VOLUME 06 | ISSUE 10A

digit FastTrack
YOUR HANDY GUIDE TO EVERYDAY TECHNOLOGY

to HTML5

Windows Internet Explorer 9

HTML5 has managed to garner interest, similar to web 2.0. Here's all the information you need to experience the future of the web

- Tenets of a modern browser
- HTML5 – an Introduction
- HTML5 Features
- Building Pinned Sites
- CSS 3
- JavaScript
- Building interoperable sites
- HTML5 Canvas and SVG
- Multimedia experiences with HTML5 audio and video

A 9 9 Media Publication

FAST TRACK to HTML

powered by

digit

thinkdigit.com

CHAPTERS

HTML5
OCTOBER 2011

CREDITS
The people behind this book

EDITORIAL
Executive Editor
Robert Sovereign-Smith

Asst. Editor – dev.works and Custom Publishing
Nash David

Writers
Amit Meena and Ankur Mour

DESIGN
Sr. Creative Director
Jayan K Narayanan
Art Director
Anil VK
Associate Art Director
PC Anoop
Visualisers
Prasanth TR & Anil T
Senior Designer
Baiju N V

October 2011
Free with Digit. If you have paid to buy this Fast Track from any source other than 9.9 Mediaworx Pvt. Ltd., please write to editor@thinkdigit.com with details

COVER DESIGN: PRASANTH TR

INTRODUCTION

Over the last few years, there have been more changes in the browser space and the internet than ever before. For customers, the web has moved from being a utilitarian tool to a personalised experience. The increase in social networking usage, the proliferation of video on the web, and the increasing number of rich web and mobile applications connecting and engaging users online have made people come to expect more from their online experience.

The rich graphics and interactive capabilities once exclusively associated with PC or mobile applications are increasingly expected from your favourite web site, be it an email service, photo site, social network, or a news site. Yet, these kind of immersive experiences do not occur on the web today.

Users have one kind of experience with applications installed on their PCs and something entirely different with the web. Browsers are often associated with limited performance and interactivity. The content that you care about is typically displayed in a simple and flat format.

Simply put, the web is not as nearly as fast, rich, or intuitive as it could be, and yet surfing the web is what people do most on their PCs. However, the web is about to take a significant step forward.

For a better web experience, you need a browser that is built around HTML5 and other modern web standards. For HTML5 to reach its full promise you also need a browser that is designed to take advantage of the power of your full PC. You also need to put your sites at the centre of your experience. Your favourite sites should be seamlessly integrated with Windows and look and behave just like a native application experience. Together, these elements promise to unlock what's next on the web.

TENETS OF A MODERN WEB BROWSER

We look at an HTML5 compatible browser such as Internet Explorer 9, understand it to a specific detail, and analyse what the future promises...

There are over a billion Windows customers in the world today. Most of them spend more time browsing the web than any other activity on their PC. These are the customers for whom Internet Explorer has been developed, so they can get more done within the browser in a safe, secure and high performance way.

The focus is to make sure that the number one activity that customers do – browsing the web, is as compelling an experience as native Windows applications installed on their PCs.

Before we dive into understanding HTML5, it is important to understand the components of a modern web browser and how HTML5 fits in.

What makes a modern browser

First, developers should take advantage of HTML5 today. The goal is "same markup", which means that developers can now move away from creating multiple versions of single sites and from creating browser prefixes to go with specific browsers. As developers have rallied around the promise of a new class of rich HTML5 web applications, Microsoft worked with standards bodies like the World Wide Web Consortium (W3C) to bring these standards to the marketplace. With Internet Explorer 9, the focus is on giving developers a stable platform with site-ready standards so that they can use HTML5 today. Internet Explorer 9 is the most standards-compliant browser ever shipped by Microsoft.

Second, to transform the flat web experience of today, the power of modern hardware must be unleashed on the web. Rich applications require robust computing capabilities that span software and hardware. As web applications become increasingly complex and graphically based, and as customers spend more and more time using these applications, browsers need to be able to take advantage of computing power offered with hardware and operating systems. This is how native applications have used Windows for years. Today, the average browser uses only 10 percent of the computing power that is available in a modern PC. Internet Explorer 9 unlocks the remaining 90 percent.

Third, people go online to get to the websites and applications they care about most—to get news about the sports that they care about, to learn about community events that matter to them, and to connect with the people that enrich their lives. Browsers can enhance that online experience allow websites to come forward and shine. Internet Explorer 9 has a clean user experience that puts the web first and lets your sites shine.

And finally, the focus on security and privacy needs to continue to be second to none. Because the web still isn't as secure or private as it should be, Internet Explorer 9 continues its industry leadership in being the world's most trusted browser and providing you with a robust set of features to help protect you and your privacy while you are online.

What's new in Internet Explorer 9

Internet Explorer 9 makes your web feel as native as the applications running on Windows.

- **Fast: Internet Explorer 9 is all around fast.** Part of reimagining the role of the browser to deliver immersive, compelling web experiences is rethinking the concept of speed. Today, speed is too often narrowly defined

as page load time. Tomorrow, a browser will not be able to call itself fast unless it lets people interact with graphically rich sites and applications with lightning speed. Fully hardware-accelerated graphics, text, video, and audio through Windows means that the same markup not only works throughout the web, but runs faster and delivers a richer experience. Designed to take full advantage of the power of your PC's hardware through Windows, Internet Explorer 9 delivers rich and immersive experiences that are as fast and responsive as applications installed on your PC

- **Clean: Internet Explorer 9 puts the focus on the web sites you love, with a clean experience for your web that meets you where you are.** Today, your favorite content, your sites and applications are buried behind a browser. Internet Explorer 9 reimagines the role of the browser and how people interact with websites and web applications. Internet Explorer 9 is site-centric, as opposed to browser-centric. You have the websites you love with a clean look that makes your websites shine. Streamlined notifications let you know when something has changed without getting in the way and pull you back into your web experiences. Jump Lists for Pinned Sites puts your favorite content just a click away without having to open your browser.
- **Trusted: Internet Explorer 9 helps people feel confident and in control.** The more that the web becomes part of our everyday lives, the more complex that the issues of online trust and security become. When done correctly, creating a trustworthy browser helps customers feel connected to the web, not distracted by concerns about reliability, privacy, or safety. People want to know that when they are doing something in one browser tab, they won't lose their work that is in another tab. They want to know that the sites that they visit and the files that they download aren't going to cause harm to their PCs or personal data. They also want to know that their private information is kept private, and that they are in control of the decision to keep their data private. Internet Explorer 9 is the trusted way to access the web because it has a robust set of built-in security, privacy, and reliability technologies that can help keep users safer and their browsing experience virtually uninterrupted.
- **Internet Explorer 9 is standards compliant.** Extensive support for HTML5, SVG, CSS3, Geolocation, ECMAScript5, and DOM provides a new set of capabilities that will help enable developers to write one set of markup and know that it will work and look the same in all modern browsers. Internet Explorer 9 was designed with support for industry

standards built in to help ensure that the same markup works the same with multiple browser types. We will explore some of these aspects in the later chapters

Let's look at each of these facets in detail.

a. Fast - Performance Improvements in Internet Explorer 9

Browser performance is a multi-dimensional problem. Many subsystems in the browser need to work together to display a page and allow you to interact with it. The picture below shows all the subsystems involved in rendering a page.

Browser Subsystem

- **Networking:** The networking subsystem is responsible for all communication between the client and server, including local caching of web content. The networking subsystem is generally gated on the performance of the user's network
- **HTML:** As HTML documents are downloaded from the server they're passed to an HTML subsystem which parses the document, initiates addi-

tional downloads in the networking subsystem, and creates a structural representation of the document. Modern browsers also contain related subsystems which are used for XHTML, XML and SVG documents.

- **CSS:** When CSS is encountered, whether that's inside an HTML document or a CSS document, it's passed to a CSS subsystem which parses the style information and creates a structural representation that can be referenced later.
- **Collections:** HTML documents often contain metadata, for example the information described in the document head or the attributes applied to an element. The collections subsystem is responsible for storing and accessing this metadata.
- **JavaScript:** When script is encountered, it's passed directly to the JavaScript subsystem which is responsible for executing that script. The JavaScript subsystem is probably the most well-known of the browser subsystems thanks to the visibility it has received over the last few years.
- **Marshaling:** Because most JavaScript engines are not directly integrated into the browser, there is a communication layer between the browser and the script engine. Passing information through this communication layer is generally referred to as marshaling.
- **Native OM:** JavaScript interacts with the document through the Document Object Model API's. These API's are generally provided through a subsystem which knows how to access and manipulate the document and is the primary interaction point between the script engine and the browser.
- **Formatting:** Once the document is constructed, the browser needs to apply the style information before it can be displayed to the user. The formatting subsystem takes the HTML document and applies styles.
- **Block Building:** CSS is a block based layout system. After the document is styled, the next step is to construct the rectangular blocks that will be displayed to the user. This process determines things like the size of the blocks and is tightly integrated with the next stage - layout.
- **Layout:** Now that the browser has styled the content and constructed the blocks, it can go through the process of laying out the content. The layout subsystem is responsible for this algorithmically complex process.
- **Rendering:** The final stage of the process occurs inside the rendering subsystem where the final content is displayed to the user. This process is often referred to as "drawing to the screen" and may occur on the CPU, the GPU, or a combination of both.

Each site has very different performance characteristics. Some spend a lot of time executing JavaScript, others spend a lot of time in marshaling, and some spend significant time in layout and rendering. To make a browser that is all-around fast, you have to understand how websites are built. For Internet Explorer 9, the development patterns used to build more than 7,000 real-world websites were studied, paying special attention to the top 1,000 websites worldwide. The analysis covered more than 50 dimensions to build a deep understanding of the patterns of those sites and took into account a range of performance characteristics including: time spent in different subsystems, use of common development frameworks like jQuery and DOJO, what analytics sites were using, how many script files sites were using, source code size, and also which features were being used.

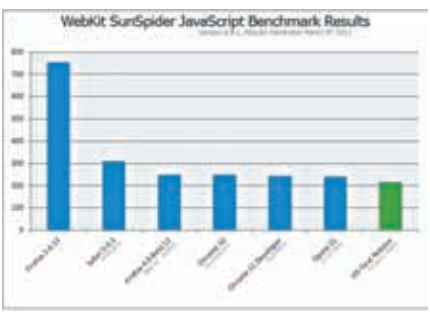
Hardware-accelerated graphics

Today, when you browse the web, you use about 10 percent of the power of your PC. Internet Explorer 9 unlocks the other 90 percent. Many browsers do this by taking a lowest common denominator approach to performance and minimizing PC capability. Internet Explorer 9 takes full advantage of the operating system and hardware to fully accelerate rendering of webpages. As an example, we use the graphics processing unit (GPU) when dealing with text, graphics, and new HTML5 technologies like Video, SVG, and Canvas.

Over the last 10 years, the computing power of the GPU has grown exponentially and today's GPUs can achieve more than one teraflop of computation. When you compare that to CPUs which have barely achieved 100 gigaflops of computation, you can see how much computation power can be unlocked through hardware.

Chakra, a brand new JavaScript engine.

Based on data, we know that most PCs have more than 2 CPU processor cores and Chakra, the new JavaScript engine in Internet Explorer 9, is optimized to take advantage of those multiple cores by compiling the JavaScript into highly



WebKit SunSpider JavaScript Benchmark

efficient machine code in the background, while interpreting the JavaScript in the foreground. Once compilation is finished, Chakra switches to the compiled and significantly faster machine code optimized for that PC. Chakra has been systematically tuned to focus on patterns that are found in real world sites.

Although not a goal for the performance work on Chakra, background compilation and many other improvements mean that Internet Explorer 9 scores very well in popular JavaScript benchmarks like WebKit's SunSpider. According to WebKit's SunSpider test, IE9 is the fastest JavaScript browser as of the Internet Explorer 9's release, when the benchmark was last used. However, measuring the performance of a browser by looking at small parts of individual subsystems is inherently flawed. Instead, better benchmarks represent real coding patterns and engage multiple subsystems at once.

New layout system

Internet Explorer 9 has an all new layout system that is optimized to reduce memory use, particularly for HTML5 websites, which will place even more demands on browsers. We find that for large and visually complex websites, the new layout engine reduced memory uses in some cases by as much as 50 percent. In addition, the new layout system forms the foundation for the graphically rich HTML5 applications of tomorrow to ensure that HTML5 capabilities such as SVG are not only fast, but are also able to scale to the needs of developers.

Add-on Performance Advisor

An addition to the feature in Internet Explorer 8 that exposed add-on load time, Add-on Performance Advisor notifies you when add-ons are slowing down your browsing session. You are notified if the total load time of all enabled add-ons takes more than 0.2 seconds, giving you an opportunity to make an informed decision to use the add-ons that you find valuable and disable those that are less useful or too slow.

Hang Recovery

New in Internet Explorer 9, this feature isolates the impact of a hung tab to the individual tab, so that other tabs and the overall browser continue to operate. When a website hangs because of a long running script or other operation, it causes your browser to become nonresponsive. Hang recovery in Internet Explorer 9 means you can continue browsing on

other tabs. This new feature compliments tab isolation and automatic crash recovery, which also helps to keep you browsing and prevent the loss of information.

And many more performance optimizations

The core of Internet Explorer 9 alsohas been fundamentally re-designed to be fast. This includes numerous performance optimizations, includingsuch as improved network caching algorithms, quicker webpage formatting with CSS, and ensuring that important APIs for web developers such as document.getElementByID are incredibly fast, significant scenario tuning (WAC, networking, benchmarks, bad pages, and so on), network cache improvements, compiler optimizations, and binary training. Internet Explorer 9 has decreased timer resolutions from a system default of 15.6ms to 4ms and enhanced the find-on-page performance. Internet Explorer 9 also made large working set reductions such as delayed image decoding (between 1-50MB depending on the site), reduced the InPrivate data collection footprint by 5MB, and reduced the size of the binary by over 1MB.

b. Site-Centric Design Makes the Web Shine in Internet Explorer 9

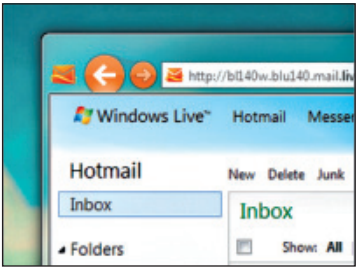
Windows Internet Explorer puts the focus on the web with a clean look and feel that makes your websites shine. With a site-centric approach, Internet Explorer 9 delivers one-click access to websites pinned directly to your task bar, fewer interruptions and navigation that works seamlessly and intuitively with the rest of Windows 7. Simply put, Internet Explorer 9 focuses on the web, not the browser.

Clean browser interface

The role of the browser is not to simply get out of the way, but to bring sites forward. In Internet Explorer 9 the simplified yet enhanced interface puts the focus on the content of each website.

Internet Explorer 9 lets your website shine

By default, only the controls essential for browsing are in the browser frame, quietly in glass, letting you browse and experience all that your sites can offer.



Primary colours and icon from the website integrate into the browser

Characteristics of each website are reflected throughout the web surfing experience, allowing you to focus on and be more immersed in the site's content. Elements of each website are integrated into the browser. If a site is pinned to the taskbar, when the site is launched, the navigational controls integrate the site's icon and primary color.

The New Tab page has been revamped in Internet Explorer 9 to intelligently display the websites that you like most and put them one click away, so that navigation is simple and easy. New users can get started quickly and returning users get meaningful suggestions and information that helps them decide what to do next as they browse.

Each site's icon and primary color is used on the New Tab page to help you identify sites more easily. This is helpful when there are many sites to choose from. Also, a site indicator shows if the site is visited often or not. From the New Tab page, you can also reopen closed tabs, reopen the last browser session, clear the site indicators, or start Microsoft InPrivate Browsing

Enhanced Tab Navigation

Pinned Sites

Most people visit a small number of websites daily, often several times a day.

You can pin sites by clicking the icon to the left of the web address and dragging it to the taskbar. After a site is pinned, it shows up as its own thumbnail, separate from Internet Explorer. When a site is launched from the taskbar, the browser frame and navigational controls integrate the site's icon and primary color, emphasizing the site and providing an even more site-focused experience

Internet Explorer 9 also provides an integrated Windows navigation experience with websites that support Jump Lists and thumbnail preview controls in the taskbar.

c. The Browser People Can Trust

Windows Internet Explorer 9 includes built-in default settings that can help keep your computer protected from the first time you launch the browser and makes your browsing reliable, safe and private

Tracking Protection

Tracking Protection helps you stay in control of your privacy as you browse the web. Some of the content, images, ads, and analytics that you see on the websites have the ability to potentially track your behavior across multiple sites. Tracking Protection provides you an added level of control and choice about the information that third-party websites can potentially use to track your browsing activity.

To use this functionality, you simply have to add a Tracking Protection List from one of the Tracking Protection List providers. These Tracking Protection Lists contain domains which Internet Explorer will block as well as domains Internet Explorer will not block. As you browse to different sites, Internet Explorer helps ensure that personal information about you, such as your IP address or the site you are currently viewing, is not sent to the domains that are blocked based on the heuristics of the list. Tracking Protection stays on until you decide to turn it off.

SmartScreen Filter

Internet Explorer 9 invests heavily in industry-leading Microsoft SmartScreen Filter and the back-end reputation systems that support it. SmartScreen is a dynamic security intelligence and safety service designed to help protect Internet Explorer users from phishing attacks and malicious software.

SmartScreen Application Reputation

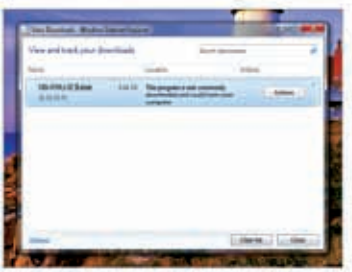
SmartScreen Application Reputation is a groundbreaking browser feature that uses reputation data to remove unnecessary warnings for well-known files and to show more severe warnings when the download is at higher risk of being malicious. Many people are often conditioned to ignore the generic warnings that are shown for every download, such as: “This file type can

harm your computer. Are you sure you want to run this file?" This same warning is presented whether the file is a common program or a piece of malware. Other browsers leave it up to you to decide if a program is safe to download and run from the Internet. Internet Explorer 9 is the only browser that uses Application Reputation to help you make safety decisions.

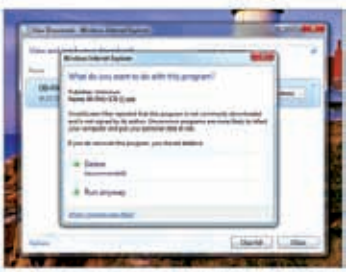
What happens if a download doesn't have a positive reputation?

A downloaded program is first checked against the Application Reputation service. For the vast majority of downloads, the downloaded program will have a positive reputation and the program can be downloaded and run with no browser warnings. However, in the rare case that a download does not have a positive reputation, you are warned and can choose to delete the program immediately or pick an option from the Actions button found on the notification bar or in the Download Manager. This warning gives you additional information about the risk of your activity so you can choose to proceed with the download or not.

Uncommon download warning in notification bar



Uncommon download warning in Download Manager



Uncommon download warning Action Button with user choices

SmartScreen URL filtering improvements

The SmartScreen URL filter continues to be a key safety asset of Internet Explorer 9.

The new Download Manager blocks downloads from known malicious websites. When a malicious download URL is detected, a warning is shown in the new notification bar and in the Download Manager. At this point,

you can continue the download—otherwise the download is cancelled and removed automatically. The Download Manager also provides you with the status of downloads, a secure area for downloaded files, and the final location where downloads are stored.

Download Manager establishes a folder for downloads so that you don't have to pick a folder each time you download a file. It's an easy way to scan and interact with file downloads.You also get to see the speed of each download. That way you're able to pause a less important download so higher priority downloads can be completed faster.

If a download is interrupted, the files can be resumed on the next launch of Internet Explorer 9. So if you encounter a network connection problem or have to shut down your computer, you easily can pick up right where you left off.

The SmartScreen block page has been updated to be clear when hosted content is malicious rather than the hosting website.

InPrivate Browsing

Sometimes customers don't want to leave a trace of their web browsing activity or browsing history on their computers. Microsoft InPrivate Browsing helps prevent browsing history, temporary Internet files, form data, cookies, usernames, and passwords from being retained by the browser. You can start InPrivate Browsing from the New Tab page, from the Internet Explorer Jump List, or by selecting InPrivate Browsing from the Safety menu.

Greater protection against emerging threats

Cross-site scripting attacks are a leading online threat. Their aim is to capture keystrokes and record sign-in information for your accounts. You might receive an email message that contains a web address that has been tampered with. When you click the link, you are directed to a legitimate website that has been compromised to contain malicious content that can capture keystrokes and record your sign-in and password information. Just like with Internet Explorer 8, Internet Explorer 9 includes a cross-site scripting filter that can detect these types of attacks and disable the harmful scripts. This protection is always on automatically.

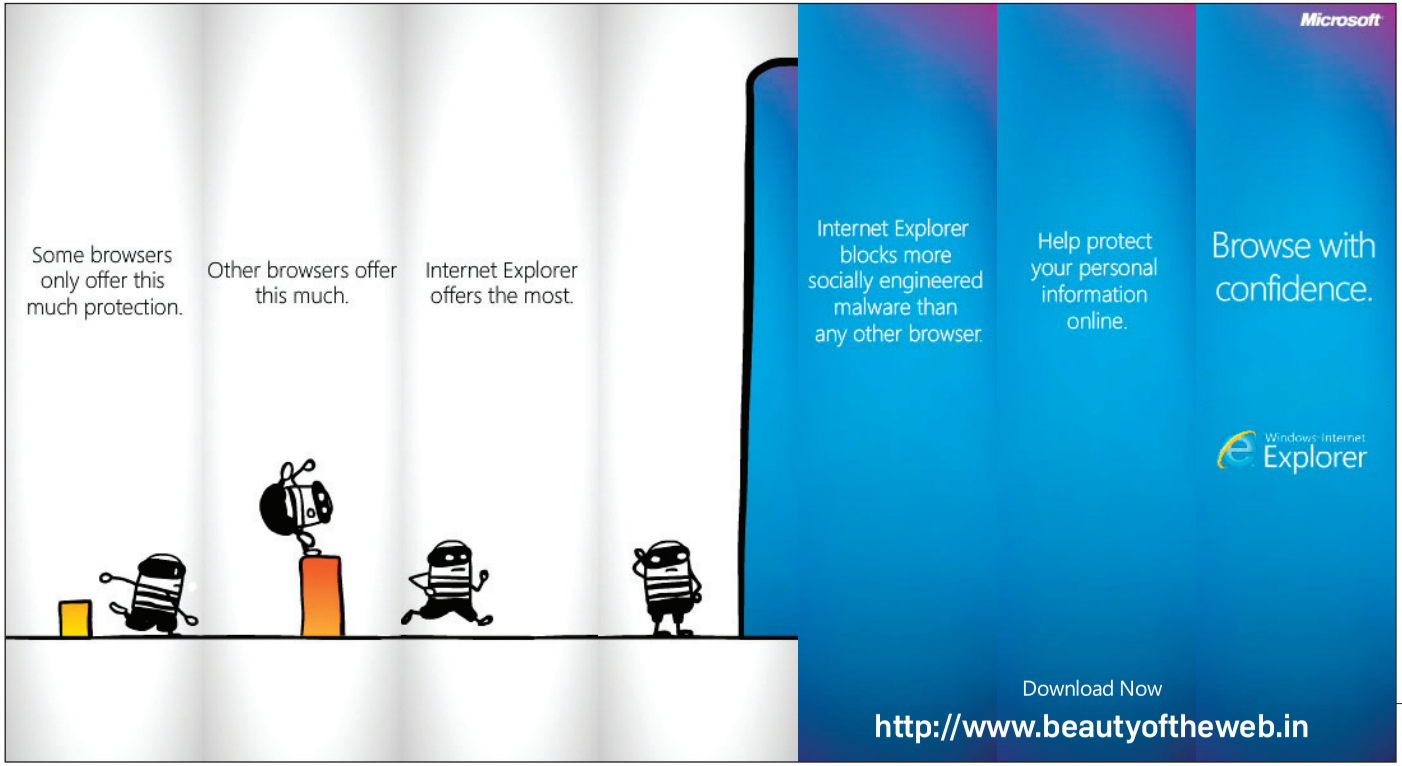
Domain highlighting

Internet Explorer 9 can help you avoid deceptive sites and can give you peace

of mind. As with Internet Explorer 8, Internet Explorer 9 takes domain names which appear in the address bar and highlights them in black, while the rest of the web address is displayed in gray text. This makes it easier to confirm the identity of the sites that you visit and helps to alert you about deceptive websites with misleading addresses. Internet Explorer 9 recognizes EV certificates for businesses that have completed this process, and visually represents them by coloring the Address Bar in green. Clicking on the security status bar displays the security report for the site.

Conclusion

Raise your expectations for the web and help the web go native. With Internet Explorer 9 you can take full advantage of Windows and current hardware to help enhance the capabilities of the web. Get an experience that makes the web feel as native as your favorite applications on your PC and enjoy a wave of immersive, beautiful websites. Developers can now create with Internet Explorer 9 and unlock the Beauty of the Web with Internet Explorer 9 (*www.beautyoftheweb.com).*


Some browsers only offer this much protection.
Other browsers offer this much.
Internet Explorer offers the most.
Internet Explorer blocks more socially engineered malware than any other browser.
Help protect your personal information online.
Microsoft
Browse with confidence.
Windows Internet Explorer
Download Now
http://www.beautyoftheweb.in

HTML5 – AN INTRODUCTION

We've heard a lot about HTML5. Whether you're wondering if it's a development platform, or you are already working on it, this chapter will help you understand its concepts better.

HTML5 is brand new. Indeed, it is far from even being finished yet. If you were to listen to some authorities on technology, they'd tell you it'll probably not be ready for another decade! Down to its fundamentals, HTML5 is a language for structuring and presenting content for the world wide web. It is the fifth revision of the HTML standard (originally created in 1990 and most recently standardised as HTML4 in 1997). As of September 2011, HTML5 is still under development and expected to be finalised by 2016. However many components with HTML5 spectrum are ready for use already.

What you should know...

Before we get down to the history of HTML5 and how we got where we are, here are five things that you should know about HTML5:

Not One Big Thing

HTML5 is now popularly used as an all-inclusive term. Typically it refers to a collection of new web specifications that enable next generation web applications. It can include anything from the true W3C HTML5 specification, CSS3, ECMAScript 5 and much more. When looked at together, they provide us as developers with new support for rich graphics and media, new JavaScript and DOM functionality to provide advanced programmability and standardise behaviour within the browsers.

HTML5 is a collection of scores of independent individual features. So it is wrong to consider it one giant entity. There is no one thing called "HTML5 support". If you want to use HTML5 in your browser, you will have to check for support for individual features of HTML5, be it canvas, video, audio or geolocation.

Think of HTML as comprising tags and angle brackets. The interaction of these angle brackets with JavaScript, through the Document Object Model (DOM), is also defined in the HTML5 specification. The easiest way to sum up the components of HTML5 is as follows:

HTML5 = HTML5 + ECMAScript + CSS 3 + SVG + GeoLocation + WebApps

As you can see, the HTML5 on the left refers to loose marketing and technology jargons. On the other hand, on the right side, it refers to the new markup, tags and features being introduced by the W3C working group.

ECMAScript is the formal body responsible for defining the language that is better known as JavaScript!

No need to start from scratch

One of the key benefits of using HTML5 is that it doesn't require you to relearn things you already know. If you're still stuck on HTML4, this is good news. If your web application worked yesterday in HTML 4, it will still work today in HTML5. Period. What HTML5 does is it gives you the tools and resources to improve your application.

Support

HTML5 is well supported by all major browsers – be it IE9, Firefox, Chrome or Safari. Even mobile browsers built into iPhone, certain Android devices and even Internet Explorer on Windows Phone Mango support HTML5. It doesn't matter which HTML5 feature you're using – designing better web forms, drawing on a canvas, playing a video or building web applications that work offline, you'll find that HTML5 is already well-supported.

It is easy to get started

HTML5 is here to make your life easier and builds on the success of HTML4. Moving to HTML5 can be as simple as changing your <doctype>. The doctype should already be on the first line of every HTML page. Previous versions of HTML defined a lot of doctypes, and choosing the right one could be tricky. Compare, for example, the following HTML4 doctypes:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

Definitely, not the easiest thing to remember and there are over ten variations of this markup! HTML5 solves this problem by defining only one doctype: `<!DOCTYPE html>` Now that's a doctype you might just remember. Upgrading to the HTML5 doctype won't break your existing markup, because all the tags defined in HTML 4 are still supported in HTML5. But it will allow you to use — and validate — new semantic elements such as <article>, <section>, <header>, and <footer>.

No escaping it

There's no way you'll be able to escape HTML5! Although it has been around for some time now, HTML5 has gained recent momentum due to a strong push from multiple vendors, including Microsoft, Apple and Google. In fact, Apple issued a public letter titled "Thoughts on Flash", where it concluded that with the development of HTML5, Adobe Flash is no longer necessary to watch video or consume any kind of web content. Definitely, HTML5 is here to stay.

History

Tim Berners-Lee invented the world wide web! Tim created the World Wide Web Consortium (W3C). Its mission was to lead the world wide web to its full potential by developing protocols and guidelines that ensure long-term growth for the web. In December 1999, HTML 4.01 was published. After this, the people who ran the W3C declared that it would be difficult to extend HTML 4 and they gave up on it for the next ten years!

Turns out that web developers got busy trying to bridge the gaps in HTML5 by rolling out custom controls plug-ins and libraries. And finally, key browser vendors such as Microsoft, Google, Mozilla and Apple got together to develop these further, finally ending up under the W3C HTML working group.

CSS and what it's all about

You might have a vague idea that it has something to do with the stylising your HTML5 page. Here's more on it.

CSS is an acronym for Cascading Style Sheets. It is a style sheet language used to describe the presentation semantics (the look and formatting) of a document written in a markup language. Although we will deal with the application of CSS to style web pages written in HTML, but the language can also be applied to any kind of XML document.

HTML elements enable web page designers to mark up a document as to its structure. The HTML specification lists guidelines on how browsers should display these elements. For example, you can be reasonably sure that the contents of a strong element will be displayed bold-faced. But you have very limited control over how your text appears. What CSS does is it puts the designer in the driver's seat. For example, CSS covers fonts, colors, margins, lines, height, width, background images, advanced positions and many other things

JavaScript / ECMAScript 5.0

JavaScript is *the* scripting language of the web and browsers implement the standards described by ECMAScript; though not all browsers implement it exactly the same way! Anyhow, JavaScript is used in billions of web pages.

It is used to add functionality, communicate with the server, validate forms, and a host of other stuff. We'll try and give you a brief idea of this scripting language in the context of HTML5.

HTML5 – FEATURES

Here's the real deal. All you can experience in HTML5 – features, functions and innovations.

Ever since its introduction, HTML has been in continuous evolution. Different features have been introduced at different points of time, some being introduced in specifications while others were introduced in software releases.

Components of HTML5

The design principles of HTML5 have been spelled out in the WHATWG specification as follows:

Compatibility

At the core of HTML5, is the desire to keep everything working smoothly. The motto is evolution not revolution.

Utility

The HTML5 specification is written based on a one-line priority order - "the

user is king." If any doubt ever arises, the specification values users over authors, over implementers (browsers), over specifiers (W3C/WHATWG), and over theoretical purity. All this makes HTML5 overwhelmingly practical, even if it's imperfect.

Interoperability

HTML5 strives for simplification and avoiding needless complexity. To achieve all this simplicity, the specification has become much bigger and detailed, spanning over 1200 pages. HTML5 is also designed to handle errors, with a variety of error handling plans.

Universal access

This principle aims to make HTML5 independent of the device or platform, support all world languages and even support users with disabilities.

The figure below shows the various classes that make up the complete HTML5 Specification. Lets examing these briefly. Keep in mind that the complete HTML5 is still a working draft and is still being discussed on the HTML Working Group and WHATWG mailing lists

HTML5 Specification

Semantics

Giving meaning to structure, semantics are front and centre with HTML5. A richer set of tags, along with RDFa, microdata, and microformats, are enabling a more useful, data driven web for both programs and your users.

Offline and Storage

Web apps can start faster and work even if there is no internet connection, thanks to the HTML5 App Cache, as well as the Local Storage, Indexed DB, and the File API specifications.

Device access

Beginning with the Geolocation API, web applications can present rich, device-

aware features and experiences. Incredible device access innovations are being developed and implemented from audio/video input access to microphones and cameras, to local data such as contacts and events, and even tilt orientation.

Connectivity

More efficient connectivity means more real-time chats, faster games, and better communication. Web sockets and server-sent events are pushing (pun intended) data between client and server more efficiently than ever before.

Multimedia

Audio and video are first class citizens in the HTML5 web, living in harmony with your apps and sites. Lights, camera, action!

3D, Graphics and Effects

Between SVG, Canvas, WebGL, and CSS3 3D features, you're sure to amaze your users with stunning visuals natively rendered in the browser.

Performance and Integration

Make your web apps and dynamic web content faster with a variety of techniques and technologies such as Web Workers and XML*HttpRequest*

CSS3

CSS3 delivers a wide range of stylisation and effects, enhancing the web app without sacrificing your semantic structure or performance. Additionally, Web Open Font Format (WOFF) provides typographic flexibility and control far beyond anything the web has offered before.

Changes to document structure

In addition to the above, there have been some changes to the document structure for HTML5:

The document structure of HTML5 is compatible with HTML4 and XHTML1 documents published on the web, but isn't compatible with some of the SGML features of HTML4. HTML5 also defines detailed parsing rules for this syntax, which is largely compatible with popular implementations.

New DOCTYPE

The HTML syntax of HTML5 requires a DOCTYPE to be specified to ensure

that the browser renders the page in standards mode. The DOCTYPE has no other purpose and is therefore optional for XML.

In HTML5, the DOCTYPE for web pages has been greatly simplified. Compare, for example, the following HTML4 DOCTYPEs:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
```

Impossible to remember, right? HTML5 neatly solves this problem as follows:

```
<!DOCTYPE html>
```

Yes, that's it! Just two words "DOCTYPE" and "html". All this is possible because HTML 5 is no longer part of SGML, but is instead a markup language all by itself.

Character declaration

Like the new DOCTYPE, the character set declaration has also been abbreviated. It used to be:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Now, it uses UTF-8 and you define it with just one meta tag:

```
<meta charset="utf-8">
```

MathML and SVG

The HTML syntax of HTML5 allows for MathML and SVG elements to be used inside a document. For example, a very simple document using some of the minimal syntax features could look like:

```
<!doctype html>
<title>SVG in text/html</title>
<p>
 A red circle:
 <svg> <circle r="50" cx="50" cy="50" fill="red"/> </svg>
</p>
```

New semantics

HTML 5 recognises that web pages have a structure, just like books have a structure. In general, web pages have navigation, body content, and sidebar content plus headers, footers, and other features. And HTML 5 has created tags to support those elements of the page.

This new markup defined by HTML5 greatly simplifies the design of your HTML pages. The new mark-up knows as semantics give meaning to the content on your page. Google had analysed millions of pages to discover the common ID names for `<div>` tags and found a huge amount of repetition. For example, web sites use DIV id="footer" to mark up footer content, HTML5 provides a sectioning element called `<footer>` that you can use in modern browsers right now.

The new HTML5 Structure

- `<section>` - It represents a generic document or application section. It can be used together with the h1, h2, h3, h4, h5, and h6 elements to indicate the document structure.
- `<header>` - It defines the header of a page or a section of the page.
- `<footer>` - It defines the footer of a page or a section of the page.
- `<nav>` - It represents the navigation on a page.
- `<article>` - It represents an independent piece of content of a document, such as a blog entry or newspaper article.
- `<aside>` - It defines extra content such as a sidebar on a page and is used to represents a piece of content that is only slightly related to the rest of the page.
- `<figure>` - defines images that annotate an article. You can use `<figcaption>` as an optional caption.

We'll now create an HTML5 page and see what it looks like. For this, we'll use the new DOCTYPE, character set, and semantic markup elements—in short, the new sectioning content.

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" >
```

```
  <title>HTML5</title>
  <link rel="stylesheet" href="digit.css">

</head>

<body>

   <header>
     <h1>Header</h1>
     <h2>Subtitle</h2>
     <h4>Fast Track to HTML5</h4>
   </header>

    <div id="container">
  <nav>
          <h3>Nav</h3>
          <a href="http://www.example.com">Introduction to
HTML5</a>
          <a href="http://www.example.com">A Brief His-
tory</a>
          <a href="http://www.example.com">New features in
HTML5</a>
        </nav>
            <section>
            <article>
              <header>
                <h1>Article Header</h1>
</header>

              <p>HTML5 is a language for structuring and
presenting content for the World Wide Web, a core technology
of the Internet. </p>
                <p>It is the fifth revision of the HTML
standard (originally created in 1990 and most recently
standardized as HTML4 in 1997) and as of August 2011 is
still under development. </p>
              <footer>
                <h2>Article Footer</h2>
              </footer>
            </article>
            <article>
```

```
            <header>
              <h1>Article Header</h1>
            </header>

              <p>Apple Inc's CEO Steve Jobs issued a public letter titled "Thoughts on Flash" where he concludes that with the development of HTML5, Adobe Flash is no longer necessary to watch video or consume any kind of web content. </p>
            <footer>
              <h2>Article Footer</h2>
            </footer>
          </article>
        </section>
          <aside>
          <h3>Aside</h3>
          <p>HTML5: There is no Escaping it !</p>
        </aside>
          <footer>
          <h2>Footer</h2>
        </footer>
  </div>
</body>
</html>
```

Without styles, the page would be pretty dull to look at,. So let some of the CSS code to jazz the content.

Document representation

Unlike previous versions of HTML and XHTML which are defined in terms of their syntax, HTML 5 is being defined in terms of the Document Object Model (DOM). It is inspired by the tree representation, which is used internally by browsers, in order to represent documents. The principle advantage of using DOM is that the language itself can be defined independently of the syntax. There are primarily two syntaxes that can be used to represent HTML documents: the HTML serialisation (known as HTML 5) and the XML serialisation (known as XHTML 5).

New elements

HTML5 introduces many new markup elements, which it groups into seven different content types.

- **Embedded** - Content that imports other resources into the document. For example, <audio>, <video>, <canvas>, and <iframe>.
- **Flow** - Elements used in the body of documents and applications, for example <form>, <h1>, and <small>.
- **Heading** - Section headers, for example <h1>, <h2>, and <hgroup>.
- **Interactive** - Content that users interact with, for example <audio> or <video controls>, <button>, and <textarea>.
- **Metadata** – Elements, commonly found in the head section, that set up the presentation or behaviour of the rest of the document, for example <script>, <style>, and <title>.
- **Phrasing** - Text and text markup elements, for example <mark>, <kbd>, <sub>, and <sup>.
- **Sectioning** - Elements that define sections in the document, for example <article>, <aside>, and <title>.

New Form Types

HTML 5 supports all the standard form input types, but it adds a few more. The idea of these new types is that the user agent can now submit a defined format to the server. It gives the user a better experience as his input is checked before sending it to the server meaning there is less time to wait for feedback.

- datetime
- datetime-local
- date
- month
- week
- time
- number
- range
- email
- url

New attributes:

HTML5 has introduced several new attributes to various elements that were already part of HTML4:

WebForms Attributes -

- A new autofocus attribute can be specified on the input select, tex-

tarea and button elements. It provides a declarative way to focus a form control during page load. Using this feature should enhance the user experience as the user can turn it off if the user does not like it, for instance.

- A new placeholder attribute can be specified on the input and textarea elements. It represents a hint intended to aid the user with data entry.

```
<input type=email placeholder="a@b.com">
```

- The new form attribute for input, output, select, textarea, button, label, object and fieldset elements allows for controls to be associated with a form. These elements can now be placed anywhere on a page, and still be associated with a form.

```
<label>Email:
 <input type=email form=foo name=email>
</label>
<form id=foo></form>
```

- The new <required> attribute applies to input, select and textarea. It indicates that the user has to fill in a value in order to submit the form.

```
<label>Color: <select name=color required>
 <option value="">Choose one
 <option>Red
 <option>Green
 <option>Blue
</select></label>
```

- The input element has several new attributes to specify constraints: autocomplete, min, max, multiple, pattern and step.

Other attributes

- The script element has a new attribute called async that influences script loading and execution.
- The html element has a new attribute called manifest that points to an application cache manifest used in conjunction with the API for offline Web applications.
- The link element has a new attribute called sizes. It can be used in conjunction with the icon relationship (set through the rel attribute; can be used for e.g. favicons) to indicate the size of the referenced icon.
- The ol element has a new attribute called reversed. When present, it indicates that the list order is descending.
- The iframe element has three new attributes called sandbox, seamless,

and srcdoc which allow for sandboxing content, e.g. blog comments.

Elements Removed

There are also some elements in HTML 4 that will no longer be supported by HTML 5.

- acronym
- applet
- basefont
- big
- center
- dir
- font
- frame
- frameset
- isindex
- noframes
- noscript
- s
- strike
- tt
- u

APIs

HTML5 introduces a number of APIs that help in creating Web applications. These can be used together with the new elements introduced for application. Some of the most important API's are:

- Geolocation API defines a high-level interface to location information associated only with the device hosting the implementation, such as latitude and longitude. The API itself is agnostic of the underlying location information sources. Common sources of location information include Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs, as well as user input. No guarantee is given that the API returns the device's actual location.
- IndexedDB API is a W3C draft Web specification for the storage of large amounts of structured data in the browser, using indexes that allow for high performance searches on this data. The IndexedDB API is currently being standardized by the W3C Web Applications Working Group. IndexedDB can be used for browser implemented functions like

bookmarks, and as a client-side cache for web applications such as email.

- WebSockets API is a technology designed to simplify much of the complexity around bi-directional, full-duplex communications channels, over a single Transmission Control Protocol (TCP) socket. It can be implemented in web browsers, web servers as well as used by any client or server application. The WebSockets API is currently being standardized by the W3C Web Applications Working Group and the protocol is being standardized by IETF Hypertext Bidirectional (HyBi) Working Group.
- FileAPI is a W3C draft Web specification for representing file objects in web applications, as well as programmatically selecting them and accessing their data. The FileAPI is currently being standardized by the W3C Web Applications Working Group.
- MediaCapture API The W3C HTML Speech Incubator Group is considering the feasibility of integrating speech technology in HTML5. At this stage, the work is very experimental and the group is discussing a number of different proposals.
- Selector APIs introduces new simple ways to find elements in your page DOM. Some previous JavaScript Methods that allowed developers to make a few calls to find specific elements in the page includes –

Selector API

- **getElementById()** - Returns the element with the specified id attribute value
- **getElementsByName()** - Returns all elements whose name attribute has the specified value
- **getElementsByTagName()** - Return all elements whose tag name matches the specified value

With the new Selectors API, there are now more precise ways to specify which elements you would like to retrieve without resorting to looping and iterating through a document using standard DOM. New QuerySelector methodsinclude –

- **querySelector()** - Return the first element in the page which matches the specified selector rules(s)
- **querySelectorAll()** - Returns all elements which match the specified rule or rules

console.log API has become the de facto logging standard for JavaScript developers. Many browsers offer a split-pane view that allows you to see

messages logged to the console, which is incidentally much better than making a call to alert(), since it does not halt program execution.

Newer browsers now have a native implementation of JSON to speed up parsing and serializing. The native JSON object is specified as part of the ECMAScript 5 standard covering the next generation of the JavaScript language.

Browser Support

An important question to address here is how to identify if your browser supports that HTML5 feature – be it canvas, video or form attributes.

When your browser renders a web page, it constructs a Document Object Model (DOM), a collection of objects that represent the HTML elements on the page. Every element — every <p>, every , every — is represented in the DOM by a different object.

All DOM objects share a set of common properties, but some objects have more than others. In browsers that support HTML5 features, certain objects will have unique properties. A quick peek at the DOM will tell you which features are supported.

There are four basic techniques for detecting whether a browser supports a particular feature. From simplest to most complex:

1. Check if a certain property exists on a global object (such as window or navigator).
2. Create an element, then check if a certain property exists on that element.
3. Create an element, check if a certain method exists on that element, then call the method and check the value it returns.
4. Create an element, set a property to a certain value, then check if the property has retained its value

For detection of HTML5 features your best tool is Modernizer. Modernizr is a small JavaScript library that detects the availability of native implementations for next-generation web technologies, i.e. features that stem from the HTML5 and CSS3 specifications. Many of these features are already implemented in at least one major browser (most of them in two or more), and what Modernizr does is, very simply, tell you whether the current browser has this feature natively implemented or not.

BUILDING PINNED SITES

The web you love, is just one click away. Now you can pin your favourite sites to the windows 7 taskbar. Grab it. Pin it. Love it.

In Windows 7, the taskbar shows both actively running applications and shortcuts to applications that you use frequently. By pinning an application to the taskbar, you can quickly find and launch the applications you care about most. Using Internet Explorer 9, you can also pin web sites to the taskbar, which enables you to launch sites the same way you launch other applications in Windows.

Pinned sites unlock a new set of tools – Favicons, Notifications, Jump Lists, and Thumbnail Toolbars. Pinned Sites feature a site-enhanced user interface, with an enlarged shortcut icon, with Back and Forward buttons and other interface elements that can be unified with the brand and overall appearance of the site.

With just a few lines of HTML or JavaScript code, you can use these tools to take advantage of the full power of the PC and extend the web experience beyond the browser and onto the Windows 7 taskbar.

Pinned sites are only supported in Windows 7. To pin a site, users can do one of the following to pin a site to the taskbar:

- "Tear" off a tab by clicking and dragging it to the taskbar.
- Drag the favorites icon (favicon) from the address bar in Internet Explorer to the taskbar.
- Drag an image in the web site to the taskbar.

After a site is pinned, the site renders with custom enhancements – for instance, the Back and Forward buttons match the color scheme of the web site. Sites can also be pinned to the Start menu.

The following graphic shows the taskbar elements in Windows 7.

Create a basic pinned site

Web developers can use the msSiteMode APIs to integrate pinned sites with the Windows 7 Taskbar. These APIs are divided into two groups:

FEATURE	DESCRIPTION
Taskbar button	When you click a Pinned Site button on the taskbar, the website launches in a new browser window that has been customized specifically for the site. The site icon is visible on the taskbar button.
Overlay icons	A 16x16 pixel notification icon that appears on top of the normal site icon. This notification indicates that something has changed in the website, such as new mail arriving, receiving a chat request, or being outbid in an auction. The notification is visible only when the site is running.
Jump List	The Jump List can contain static and dynamic tasks that act as shortcuts into the Pinned Site. A user can further customize their experience by pinning these items to the Jump List.
Jump List category	A Pinned Site can create and fill a single custom Jump List category by using script in the webpage. This list is dynamic.
Jump List tasks	In addition to the Jump List category, a Pinned Site can predefine a list of tasks in metadata. This list is static.
Thumbnail toolbar	A Pinned Site can create up to seven buttons that appear on the taskbar thumbnail window. These buttons can be visible or hidden, active or inactive, or toggle between different button states.

1. Site properties which apply to any user visiting the web site (these are implemented using HTML).
2. Information that is dynamic in nature or specific to an individual user visiting the web site (these are implemented using JavaScript).

Basic Pinned sites contains properties which apply to any user and explain how to implement these using HTML.

These site properties can be implemented by web developers using meta tags that live inside the head block of your HTML. The site properties that you can implement using the meta tag are explained below:

Application name

The application name will be used as the name of the application button on the taskbar, and it is appended to the window title for any pinned site window. If an application name is not given, the HTML document's Title element is used instead.

You can set the application name like this, substituting in the appropriate content for your site:

- `<meta name="application-name" content="ThinkDigit"/>`

Once your site is pinned, right-click on its icon in the Taskbar. The first link in the bottom group of links should read "ThinkDigit" and if you click it, it should open the web site.

Set the starting page

The Start Page is set using a meta tag with `name="msapplication-starturl"`. The "msapplication-starturl" metadata contains the root URL of the application. The start URL can be fully qualified, or relative to the current document. Only HTTP and *HTTPS* protocols are allowed. If this element is missing, the address of the current page is used instead.

- `<meta name="msapplication-starturl" content="http://www.thinkDigit.com" />`

Tooltips

You can also pin sites to the Start Menu or Desktop. The "Desktop Tooltips" functionality won't show up when you pin a web site to the taskbar, but you will see it as a tooltip if you pin the site to the desktop.

The syntax is as follows (again, choose your own content):

- `<meta name="msapplication-tooltip" content="ThinkDigit – Your best Tech Resource Ever" />`

To pin to the Desktop, drag the favicon from the address bar in Internet Explorer onto your desktop. This will create a link on your desktop. Hover your mouse pointer over the icon on the desktop.

Constrain the initial size of the browser window

The pinned site window size defines the size (width and height, in pixels) of the pinned site when it is first launched by the user. The syntax is:

```
<meta name="msapplication-window"
content="width=800;height=600" />
```

The initial size of the pinned site's window will now be 800x600.

Back and Forward Navigation Button Colors

This defines the color of the "Back" and "Forward" navigation buttons in the pinned site browser window. Any named color or hex color value as defined by CSS Level 3 is valid. If this meta tag is omitted, the navigation button color will be based on the favicon.

```
<meta name="msapplication-navbutton-color"
content="orange" />
```

Now the navigation buttons for the pinned window should be orange, as below:

Finally, let's put this all together with an example. Here is the HTML used to define thinkdigit.com. Feel free to pin this site and then see how the various site properties are exposed in the pinned site.

```
<html>
<head>
    <title>ThinkDigit Ie9 Pinned Site Sample</title>

    <!— All other Meta Tags go here -->


    <!-- Pinned site functionality -->
<meta name="application-name" content="ThinkDigit"/>
<meta name="msapplication-starturl" content="http://www.
thinkDigit.com" />
<meta name="msapplication-tooltip" content="ThinkDigit –
```

```
Your best Tech Resource Ever" />
• <meta name="msapplication-window"
content="width=800;height=600" />
• <meta name="msapplication-navbutton-color"
content="orange" />
•
• </head>
• <body>
•     <!-- Deleted for space... -->
• </body>
• </html>
```

How to create Jump List tasks

Tasks

While a destination is a thing, a task is an action, and in this case it is a site-specific action. In the context of a Pinned Site, a task can be any common navigation or site functionality.

Some tasks are static, meaning that they are set when the site is pinned. Others, like notifications and user-specific tasks, are dynamic and change over time.

- **Static tasks are defined using meta tags.** These properties apply to any one using the site when it is pinned and appear in the category named Tasks. This category can contain up to five items. The items in this list tend to remain the same regardless of the state or status of the application. This topic explains how to add static tasks.
- **Dynamic tasks are defined using JavaScript APIs.** These tasks typically surface information that is specific to an individual user and appear in a Jump List category that you create. This custom category can contain up to 20 items, although only the last 10 items appear in the Jump List by default.

  A pinned site can only have one static list and one dynamic list at a time.

Implementing static Jump Lists

Static Jump List Tasks are also implemented using meta tags in the head block of your HTML. You must specify a name of "msapplication-task" and a content attribute that contains a trifecta of information:

name – This is the name of the task which is displayed to the user in the Jump List.

action-uri – This URI is the destination that the user will be taken to when the Jump List task is clicked.

icon-uri – This URI is a pointer to an icon (*.ico file) that will be displayed to the left of the task name in the Jump List.

Both of the URIs can be absolute or relative. For the action-uri, any destination is valid; this is not bound to the domain of the webpage like the Start URL property.

Putting this all together, here is what a single Jump List task might look like in your HTML:

```
<meta name="msapplication-task" content="name=Locations;
    action-uri=http://www.consumermate.com/Location.aspx;
    icon-uri=http://www.consumermate/favicon.ico"/>
```

In the Jump List, this task will look like the first task below:

When you click on the "Locations" task, you will be taken to *http://www.consumermate.com/Location.aspx*. The little Windows-like icon next to the "Locations" task was pulled in from *http://www.consumermate.com/favicon.ico* (the icon-uri).

Implementing dynamic Jump Lists

Jump List categories are similar to Jump List tasks, except with categories, you can define your own heading (or category) for the items in the Jump List. They can be used to surface information specific to a user, notifications, or the user's history. For example, on a news portal, the site could build up a "Recently Viewed" category in the Jump List and include perhaps the last 5 articles viewed on the site.

Let's look at a real-world example of site pinning. If you go to *www.consumermate.com* and right-click on the Taskbar icon, you will see a Jump List like this:

Now, you will notice that the Consumermate pinned site Jump List has Laptop prices in the Tasks list. This takes you straight to the prices without having to navigate to the page from the address bar.

Now, let's look at how to code a Jump List category. First, we need to create a new Jump List category. To do that, use the msSiteModeCreateJumpList method in JavaScript. It takes a single parameter, which is the name of the Jump List category that will be displayed to the user.

```
window.external.msSiteModeCreateJumplist('Resources');
```

Next, we need to create items in this Jump List category. This is done by the `msSiteModeAddJumpListItem` method. This method has 3 required parameters, which are the same trifecta as when we were putting items in our Jump List tasks:

name – This is the name of the Jump List item which is displayed to the user in the Jump List.

action-uri – This URI is the destination that the user will be taken to when the Jump List item is clicked.

icon-uri – This URI is a pointer to an icon (*.ico file) that will be displayed to the left of the item name in the Jump List.

You will probably want to add multiple items to your Jump List category, using the same code below with different content:

```
window.external.msSiteModeAddJumpListItem('Twitter',
    'http://www.twitter.com/consumermate,
   'http://a3.twimg.com/a/1294164987/images/iefavicon.ico');
```

Next, we have to call msSiteModeShowJumpList to display the Jump List with our new items.

```
window.external.msSiteModeShowJumplist();
```

Finally, if you ever want to remove your custom Jump List, you can do this by calling the msSiteModeClearJumpList method.

```
window.external.msSiteModeClearJumplist();
```

Putting it all together, the JavaScript might look like this:

```
// Jump list categories
if (window.external.msIsSiteMode()) {
     window.external.msSiteModeCreateJumplist('Name of
the Category');
     window.external.msSiteModeAddJumpListItem('Name of
```

```
Item', 'http://www.URLLink1.com', 'http://www.URLLink1.com/favicon.ico');
        window.external.msSiteModeAddJumpListItem('Twitter', 'http://www.twitter.com/consumermate, 'http://a3.twimg.com/a/1294164987/images/iefavicon.ico');
        window.external.msSiteModeAddJumpListItem('Internet Explorer 9', 'http://bit.ly/b9HBZu', 'http://Microsoft.com/ie/favicon.ico');
        window.external.msSiteModeShowJumplist();
        //window.external.msSiteModeClearJumplist();
    }
```

The first if statement checks to see if the current page was launched as a pinned site using the msIsSiteMode method. (If the page is just running within the browser and isn't pinned, none of this functionality will be exposed anyway.)

Create advanced Pinned Sites

Implementing Overlay icons

Overlay icons are used to surface notifications or status to the user. While the pinned site is running, it has the ability to display a small icon overlaid on the Taskbar button, to draw the user's attention back to the site or notify the user that some event has happened. The pinned site can display multiple overlay icons to the user, but it can only display one icon at a time.

A great example is Facebook. If you pin facebook.com, you will see that when you have new notifications (new friend requests, messages, etc.),

The Facebook pinned site will overlay a red icon with a white asterisk to notify you.

To implement overlay icons in your own pinned sites, you will need to know how to both set and hide overlay icons.

To display an overlay icon to the user on the Taskbar button, you will use the msSiteModeSetIconOverlay method. It takes a parameter which is

a URL pointing to an icon file (*.ico); this is the icon that will be overlaid on the Taskbar button. There is also a second, optional string parameter for a description of the icon overlay, for accessibility purposes.

```
window.external.msSiteModeSetIconOverlay('Images/info.ico');
```

To clear an overlay icon from the Taskbar button, you will use the msSiteModeClearIconOverlay method.

```
window.external.msSiteModeClearIconOverlay();
```

Implementing Flashing taskbar icons

Perhaps you have an instant messaging program which flashes the taskbar button when a new message comes in? Flashing taskbar buttons are used to signal users that their attention or interaction is needed.

Pinned sites can make their taskbar buttons flash as well. To flash the pinned site taskbar button, use the msSiteModeActivate method.

```
window.external.msSiteModeActivate();
```

This will cause the taskbar button of the pinned site to start flashing. It will flash 20 times, and then the button will stay highlighted until the window comes to the foreground.

When the pinned site window comes to the foreground, the flashing will deactivate automatically.

```
Here is a complete sample:
<script type="text/javascript">
    // Flashing TaskBar
    setInterval(flashToolbar, 10000);    // calls flashToolbar every 10000 ms

    function flashToolbar() {
        if (window.external.msIsSiteMode()) {
            window.external.msSiteModeActivate();
        }
    }
</script>
```

Implementing Thumbnail toolbar icons

When you hover the mouse over a Taskbar button of a running application or website, a "thumbnail" pops up. This is a little image of the running window.

With pinned sites, you can create thumbnail toolbar buttons, which are small buttons that will display directly under the thumbnail in that popup.

Here is one example, from the site *http://www.jango.com.*

Jango is a site with free streaming music. When you play music from the site and hover over the taskbar button, a thumbnail pops up with two thumbnail toolbar buttons, to pause the music and to skip to the next song.

This is a really cool feature. You can be doing your work and have Jango playing in the background, and if a song comes on that you don't like, simply hover over the Jango Taskbar button and then click the "Next" button on the thumbnail toolbar. you don't even have to bring up the main Jango window! Thumbnail toolbars let you interact with the site without having to restore or activate the site's window.

You can have a maximum of 7 buttons in a thumbnail toolbar.

You can implement thumbnail toolbar buttons in your pinned sites using JavaScript. First, add a thumbnail toolbar button using msSiteModeAddThumbBarButton. This takes two parameters – a link to the icon resource file (*.ico) and a button name or description, which is displayed as a tooltip on hover. It then returns an integer identifier of the new button. In the code snippet below, we'll add a "Next" button.

```
btnNext = window.external.msSiteModeAddThumbBarButton('Images/next.ico', 'Next');
```

Then, you need to set up an event handler to react when that button is clicked. Use the addEventListener method which takes three parameters: the event type to respond to (you should use "msthumbnailclick" for this), the event handler function to associate with the event, and a Boolean value that specifies to add the event handler for the capturing or bubbling phase of the event (true is the capturing phase and false is the bubbling phase).

```
document.addEventListener('msthumbnailclick', onThumbnailButtonClicked, false);
```

Next, enable the thumbnail toolbar using msSiteModeShowThumbBar.

```
window.external.msSiteModeShowThumbBar();
```

Then, you need to update the button(s) using msSiteModeUpdateThumbBarButton and make visible the individual buttons. For each button, you would call this method, passing in the button ID that we got from msSiteModeAd-

dThumbBarButton, a Boolean value on whether the button should be enabled, and a Boolean value on whether the button should be visible. In the code below, we'll make our "Next" button enabled and visible.

```
window.external.msSiteModeUpdateThumbBarButton(btnNext, true, true);
```

Finally, you can optionally add button styles using msSiteModeAddButtonStyle and show the button styles using msSiteModeShowButtonStyle. This is useful if you have a play/pause button like the Jango example above. This is a single button that toggles between two states – when the music is playing, it's a "Pause" button, and when the music is paused, it's a "Play" button. You could define a "Play" style and a "Pause" style, and then just swap styles on the button when appropriate. The msSiteModeAddButtonStyle method takes 3 parameters: the button ID that we got from msSiteModeAddThumbBarButton, the URI to the icon resource file (*.ico), and an optional description of the button. It then returns the ID of the new button style. The msSiteModeShowButtonStyle method takes 2 parameters: a button ID and a style ID, to apply the specified style to the specified button.

```
stylePlay = window.external.msSiteModeAddButtonStyle(btnPlayPause, 'play.ico', 'Play');
stylePause = window.external.msSiteModeAddButtonStyle(btnPlayPause, 'pause.ico', 'Pause');

// Initially, make it a play button.
window.external.msSiteModeShowButtonStyle(btnPlayPause, stylePlay);
```

How to provide discoverabilty?

You have seen how Pinned Sites can provide a more interactive experience for your users; however, your users might not know that your site can be pinned. In this section, you learn some ways to advertise the features of your Pinned Site.

It is important to realise that Windows Internet Explorer 9 doesn't notify a user if the site can be pinned. It's up to the developer to advertise these features.

The following are just a few of the variety of ways you can alert your users to the Pinned Site features of your web site.

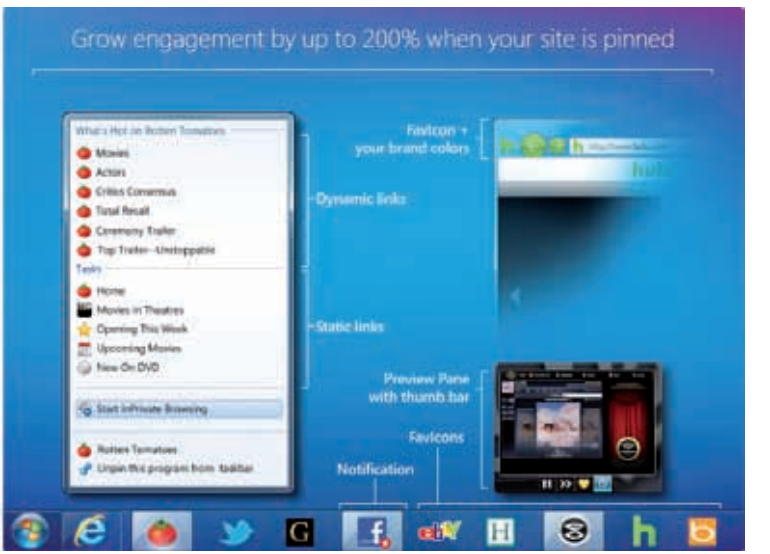
- **Fly-ins, drop-downs, banners, and div effects.** These regions detect the Pinned Site capabilities of the browser and ask the user to pin the site.

- **Drag-to-pin icons and images.** Users can pin the site by dragging any image that you provide to the taskbar.
- **First run experience.** You can easily detect when the user pins the site for the first time. This is a great opportunity to call attention to the specific features of your Pinned Site.

Summary

Web sites that implement the Pinned Site can feel more like a native Windows application. Site developers can build a more compelling web site that have higher engagement by:

- Declaring a static list of tasks for fast navigation to common destinations within a site.
- Creating a dynamic list of destinations that are personalized and relevant to the user.
- Drawing the user back to the website by flashing the taskbar button or displaying a icon overlay for notifications
- Adding remote controls and commands to the taskbar's preview window.
- Making the browser look and feel like your site by changing the color of the Back and Forward buttons.
- Promoting a high-resolution site icon that extends your site's brand outside the browser.



Implementing site pinning takes less than an hour!
For details, go to buildmypinnedsite.com

CSS 3.0

We've seen about content markup so far. With emergence of HTML5, you need to look at the latest style guidelines specified in CSS 3.0

Introduction

Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents. Cascading Style Sheets is formally described in two specifications from W3C: CSS1 and CSS2. CSS1 describes a simple formatting model mostly for screen-based presentations. CSS1 has around 50 properties . CSS2 includes all CSS1 properties and adds around 70 of its own. CSS3 now introduces modules. The advantage is that modules allows the specification to be completed and approved in segments.

It is important to note that many CSS3 modules are still in the Working Draft or Last Call stages. Until they reach the Candidate Recommendation stage, they could change specification.

CSS 3.0 Modules

In this section, we will explore in-detail some of the new CSS modules.

CSS3 Backgrounds and Borders Module

This was one of the most requested CSS module compliance. Previously, web

developers had to use complicated techniques involving multiple images or tables to create the appearance of rounded corners on layout boxes.

Notable enhancements from this module include new support for the background-clip, background-origin, and background-size properties, in addition to support for the Border-Radius and box-shadow property.

Border radius

The border-radius property is a composite property that specifies up to four border-radius properties. If values are given before and after the slash, the values before the slash set the horizontal radius and the values after the slash set the vertical radius. If there is no slash, the values set both radii equally. The four values for each radii are given in the following order: top-left, top-right, bottom-right, bottom-left. If the bottom-left value is omitted, the value is the same as the top-right value. If the bottom-right value is omitted, the value is the same as the top-left value. If the top-right value is omitted, the value is the same as the top-left value.

```
border-radius: radius | percentage
```

variable of type String that specifies or receives one or two radius values.

radius

A floating-point number, followed by either an absolute units designator (cm, mm, in, pt, or pc) or a relative units designator (em, ex, or px). For more information about the supported length units, see Values and Units.

percentage

An integer, followed by a %. The value is a percentage of, for horizontal radii, the width of the border box, or, for vertical radii, the height of the border box.

Example:

```
border-radius: 50px 50px 0 0 / 25px 25px 0 0;
```

The border-radius also has the following properties:

- border-top-left-radius: Defines the upper-left corner of a box.
- border-top-right-radius: Defines the upper-right corner of a box.
- border-bottom-right-radius: Defines the lower-right corner of a box.
- border-bottom-left-radius: Defines the lower-left corner of a box.

Example

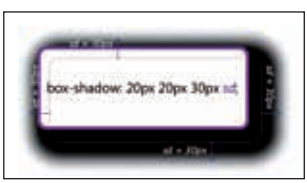
```
border-top-left-radius: 50px 25px;
border-top-right-radius: 50px 25px;
border-bottom-left-radius: 15px 30px;
border-bottom-right-radius: 15px 30px;
```

Shadows

Drop shadows are enabled by the box-shadow property. Like rounded corners, drop shadows in Internet Explorer 9 are very robust; there are several options you can specify. The order of values within the box-shadow property is as follows:

```
box-shadow: hoff voff bd sd color inset;
```

- `hoff` indicates the horizontal offset. This length value is required. It specifies how far to the right (positive value) or left (negative value) the shadow extends. See the following diagram for a visual explanation; hoff is set to 20 pixels.
- `voff` indicates the vertical offset. This length value is required. It specifies how far down (positive value) or up (negative value) the shadow extends. `voff` 20 pixels.
- `bd` indicates the blur distance. This positive length value indicates how blurred the shadow's edge is – the approximate length between the start of the blur and the end. The higher the value, the more blurred it is. See `bd` is set to 30 pixels.
- `sd` indicates the spread distance. This length value indicates how much the shadow shape expands in all directions (positive value) or contracts (negative value). Beyond the dimensions of the original shape `sd` is set to 30 pixels. The original offset border box shape is shown with a dotted line so you can better visualize the spread, which begins at the original offset box border.
- `color` indicates the color of the shadow.
- `inset` indicates the shadow is an inner shadow instead of an outer shadow. The first two values are required for the property to work, and all values must be specified in the order listed. If a color is not specified, Internet Explorer 9 uses black as the shadow color.



CSS3 2D transforms

CSS3 2D Transforms module, which enables elements that are rendered by CSS to be transformed in two-dimensional space. 2D transforms are currently supported using the following Transforms properties:

- The -ms-transform property applies a two-dimensional transformation function to an element. It contains a list of one or more transformation functions. Internet Explorer 9 supports all the transformation function specified in the CSS3 2D Transforms module Working Draft (dated-December 1, 2009).
- The following transformation functions are used with the -ms-transform property:
- The `matrix(a,b,c,d,e,f)` function specifies a 2D transformation in the form of a transformation matrix of six values (a through f).
- The `translate(tx,[ty])` function specifies a 2D translation by the vector [tx,ty], where tx is the first translation-value parameter and ty is the optional second translation-value parameter. If ty is not specified, its value is zero. (Translation-value parameters can be either percentages or lengths.)
- The `translateX(tx)` function specifies a translation by the given amount in the x direction.
- The `translateY(ty)` function specifies a translation by the given amount in the y direction.
- The `scale(sx,[sy])` function specifies a 2D scale operation by the [sx,sy] scaling vector that is described by the two parameters. If the second parameter is not provided, it takes a value equal to the first.
- The `scaleX(sx)` function specifies a scale operation by using the [sx,1] scaling vector, where sx is given as the parameter.
- The `scaleY(sy)` function specifies a scale operation by using the [1,sy] scaling vector, where sy is given as the parameter.
- The `rotate(angle)` function specifies a 2D rotation by the angle specified in the parameter about the origin of the element, as defined by the transform-origin property.
- The `skewX(ax)` function specifies a skew transformation along the x-axis by the given angle.
- The `skewY(ay)` function specifies a skew transformation along the y-axis by the given angle.
- The `skew(ax,[ay])` function specifies a skew transformation along the x- and y-axes. The first angle parameter specifies the skew on the x-axis. The second angle parameter specifies the skew on the y-axis. If

the second parameter is not given, a value of zero is used for the y angle (that is, no skew on the y-axis).

The -ms-transform-origin property establishes the origin of transformation for an element. This property is useful when you want to change the default origin (the center). The -ms-transform-origin property accepts one or two values. Each value can be a keyword, a length, or a percentage. If the -ms-transform-origin property is not set, the transform begins in the center (equal to a -ms-transform-origin value of 50% 50%).

- The first value indicates the horizontal position (the position along the x-axis), and can be negative. This value can be a length value (in any of the supported length units), a percentage (of the total box length), or one of the following three keywords: left (equal to 0% or a zero length), center (equal to 50% or half the box length), or right (equal to 100% or the full box length).
- The second value indicates the vertical position (the position along the y-axis), and can be negative. This value can be a length value (in any of the supported length units), a percentage (of the total box height), or one of the following three keywords: top (equal to 0% or a zero height), center (equal to 50% or half the box height), or bottom (equal to 100% or the full box height).

CSS3 Color Module

The CSS3 Color Module includes several new color models including alpha (transparency)-enabled models, opacity property which allows developers to control transparency at the element level, the transparency property. This module includes support for the RGBA, HSL, and HSLA color models; the opacity property; and the currentColor keyword. Internet Explorer 9 also expands support for the transparent keyword.

RGB and RGBA Color

The RGB color model has been extended to include an alpha channel, or transparency. The format of an RGBA value is `rgba(red,green,blue,alpha)`. The red, green, and blue components are identical to those of the RGB color model, and are expressed as integers or percentages. The alpha component is expressed as a value between 0.0 (completely transparent) and 1.0 (completely opaque).

For instance, to set the background color to red with 50% transparency, you can include either of the following two CSS declarations in your style sheet:

```
background-color: rgba(255,0,0,0.5);
background-color: rgba(100%,0%,0%,0.5);
```

Be aware that though RGB values support hexadecimal notation, RGBA values do not.

HSL and HSLA Color

In the hue-saturation-lightness (HSL) color model, "hue" is defined as the indicated color's angle on the color wheel (for instance, red is 0 or 360, green is 120, and blue is 240). "Saturation" and "lightness" are expressed as percentages. For instance, the following CSS declaration defines a red background.

```
background-color: hsl(0,100%,50%);
```

The HSL color model with an alpha channel, is expressed as a value between 0.0 and 1.0. For instance, the following CSS declaration defines a red background with 50% transparency.

```
background-color: hsla(0,100%,50%,0.5);
```

The opacity Property

The opacity property, enables you to control transparency at the element level. Similar to the alpha component of RGBA values, the opacity value is a number that ranges between 0.0 (completely transparent) to 1.0 (completely opaque). The opacity property is available on all elements.

The following example shows the opacity property on the color navy, with opacity values in increments of 0.2 from 0 to 1.

```
<div class="opacity _ sample">
   <div style="background: navy; opacity: 0;"></div>
   <div style="background: navy; opacity: 0.2;"></div>
   <div style="background: navy; opacity: 0.4;"></div>
   <div style="background: navy; opacity: 0.6;"></div>
   <div style="background: navy; opacity: 0.8;"></div>
   <div style="background: navy; opacity: 1;"></div>
</div>
```

This example generates the following output for a 25×125-pixel div.

CSS3 Fonts Module

Better typographic control has been a consistent feature of each new version of CSS. The CSS3

Fonts Module extends font support in CSS2.1 and redefines the behaviour of certain features at a time when the lack of an interoperable web font format frustrates many developers. Internet Explorer 9 provides full compliance with the CSS3 Fonts Module, including interoperable implementations of the @font-face rule, font-weight mapping, font-size mapping, and the font-stretch property. Internet Explorer 9 also adds support for the Web Open Font Format (WOFF), which repackages spline font (sfnt)-based font files (TrueType, OpenType, and Open Font Format fonts) by compressing each table individually using a ZIP compression format. Raw fonts (TrueType fonts with embedded permission bit that is not set) are also supported.

The font-weight property calculates font weights as specified in the CSS3 Fonts Module.

The font-size property build a scaling factor Also, keywords and HTML heading sizes are now mapped as specified in the CSS3 Fonts Module.

The font-stretch property selects a normal, condensed, or expanded face from a font family. This property is also available as a @font-face rule descriptor.

The @font-face Rule enables font linking. That is, a style sheet can reference a specific font file for the browser to download and use. For instance, consider the following markup.

```
@font-face {
   font-family: MyFont;
   src: url(http://mysite/fonts/MyFont.ttf)
      format("embedded-opentype");
}
p {font-family: MyFont, serif;
}
```

In this example, the @font-face rule instructs the browser to go to the URL specified in the src descriptor to download the font file that contains the specified font.

The unicode-range descriptor, which enables you to specify the range of Unicode characters supported by a given font. For instance, the following code example specifies the range of basic ASCII characters.

```
@font-face {
   font-family: MyFont;
   src: url(http://mysite/fonts/MyFont.ttf);
   unicode-range: U+0-7F;
```

```
}
```

The Web Open Font Format (WOFF) repackages sfnt-based font files (TrueType, OpenType, and Open Font Format fonts) by compressing each table individually using a ZIP compression format

CSS3 Media Queries Module

The CSS3 Media Queries Module specifies methods to enable web developers to correspond style sheets to precise device capabilities. For instance, a developer might want to design pages differently for mobile devices (with smaller screens, a limited color palette, low resolution, and so on) versus netbooks (with small screens, full color palettes, high resolution, and so on) versus standard PCs (with large screens, full color palettes, high resolution, and so on). The full list of media properties supported by CSS3 media queries includes width, height, device-width, device-height, orientation, aspect-ratio, device-aspect-ratio, color, color-index, monochrome, and resolution.

The following declaration is a typical media query, using the @media rule.

```
@media screen and (max-width:400px) {div {border:none;}}
```

In this case, screen indicates the target media type, and max-width is the target media property. The declaration states that the specified rules (no border on div elements) are only to be applied when the page is displayed on a screen with a width of at most 400 pixels. You can use media properties together to create even more specific queries, such as the following.

```
@media screen and (max-width:400px) and (max-height:600px)
{...}
```

This declaration applies the specified rules when the medium is a screen that has a width of no more than 400 pixels and a height of no more than 600 pixels.

CSS3 Values and Units Module

The CSS3 Values and Units Module. describes the various values and units that CSS properties accept. Also, it describes how values are computed from "specified" through "computed" and "used" into "actual" values. The main purpose of this module is to define common values and units in one specification which can be referred to by other modules. As such, it does not make sense to claim conformance with this module alone.

This module describes Textual Data Types, Numeric data types, Distance units, Angle units, Time units, frequency units, layout specific data types, Functional Notations like calculations, Cycling Values etc. Some examples are:

- deg: degrees (angle unit)
- grad: grads (angle unit)
- rad: radians (angle unit)
- turn: turns (angle unit)
- ms: milliseconds (time unit)
- s: seconds (time unit)
- rem: font size of the root element (relative length unit)
- vw: viewport width (relative length unit)
- vh: viewport height (relative length unit)
- vm: smaller of viewport width or height (relative length unit)
- ch: zero-width (width of the zero glyph in the rendering font; relative length unit)
- attr() function is used for generated content. It can be applied on any property and accept multiple arguments.
- calc(): calculates values using arithmetic operators and is usable wherever length values are allowed

CSS3 Selectors Module

The CSS3 Selectors Module specifies several additions to CSS selector syntax. They include structural pseudo-classes, the :target pseudo-class, UI element states pseudo-classes, the negation pseudo-class, and the UI element fragments pseudo-element.

Structural pseudo-classes enable selection based on extra information in the document tree that can't be selected using simple selectors or combinators.

The following selects an E element that is the root of the document.

E:root

The following selects an E element that is the n-th child of its parent.

E:nth-child(n)

The following selects an E element that is the n-th child of its parent, counting from the last one.

E:nth-last-child(n)

The following selects an E element that is the n-th sibling of its type.

E:nth-of-type(n)

The following selects an E element that is the n-th sibling of its type, counting from the last one.

E:nth-last-of-type(n)

The following selects an E element that is the last child of its parent.

E:last-child

The following selects an E element that is the first sibling of its type.

E:first-of-type

The following selects an E element that is the last sibling of its type.

E:last-of-type

The following selects an E element that is the only child of its parent.

E:only-child

The following selects an E element that is the only sibling of its type.

E:only-of-type

The following selects an E element that has no children (including text nodes).

E:empty

The target pseudo-class selects the target element of the referring URI. A fragment identifier is used to identify a location within a page, and is formed using a number sign followed by an anchor identifier—for example, *http://www.example.com/mypage.html#section_3.*

The following selects the div element of class important that is the target element of the referring URI. If the document's URI has no fragment identifier, there is no target element.

div.important:target

The UI element states pseudo-classes are used to select UI elements (form controls such as radio buttons or check boxes) that are in a certain state—enabled, disabled, or selected (checked).

The following selects an E form control element that is enabled.

E:enabled

The following selects an E form control element that is disabled.

E:disabled

The following selects an E form control element that is selected.

E:checked

The :indeterminate pseudo-class selects radio buttons and check boxes whose toggle states cannot be determined—they are neither checked (selected) nor unchecked (cleared). The following selects an E form control element whose state cannot be determined.

E:indeterminate

Note The :indeterminate pseudo-class is no longer defined in the current CSS3 specification, but is supported in many popular browsers.

The negation pseudo-class takes a simple selector as an argument to select elements that are not selected by that argument. The following selects

an E element that does not match the simple selector s:

E:not(s)

The UI element fragments pseudo-element, ::selection, is used to select any part of the page that the user has highlighted, including text within an editable text field. This pseudo-element can be applied to the color and background-color properties. The following selects any user-selected text within an E element.

E::selection

Note The ::selection pseudo-element is no longer defined in the current CSS3 specification, but is supported in many popular browsers.

Summary

CSS3 has many additions that along with HTML5 and JavaScript, will help create never seen before web experiences. Internet Explorer 9 has more support for Cascading Style Sheets (CSS) than any previous version of Windows Internet Explorer. Continuing on the work to enhance CSS support that was done in Windows Internet Explorer 8—where Internet Explorer became fully compliant with the Cascading Style Sheets, Level 2 Revision 1 (CSS2.1) specification—Internet Explorer 9 adds support for many components of Cascading Style Sheets, Level 3 (CSS3).

Internet Explorer 9 Makes Your Web Look and Perform as if it Were Native to Windows

Fast

All around fast using the power of your whole PC

Full hardware acceleration of all graphics and text

New JavaScript engine

Clean

Putting the focus on your sites

Clean web-centric UI

Seamless with Windows 7

New tab page experience

One Box

Quieter notifications

Trusted

Secure, reliable, and private

Download manager with SmartScreen protection

Hang resistance

Add-on performance protection

Interoperable

HTML5 & same markup

HTML 5 and modern web standards support

Leading with the W3C

Comprehensive standards tests

http://www.beautyoftheweb.in

INTRODUCING JAVASCRIPT

After following the specifications laid out within HTML5,, and styling in CSS 3.0, add spice to your page with JavaScript

Introduction

If HTML lends semantic structure to the content and CSS determines display then JavaScript is the component of web development that lends intelligence to the application. JavaScript is a dynamic programming language that enables rich interactive user experiences on the web. JavaScript is standards based and ubiquitous and with modern browsers, extremely fast.

JavaScript is technically a dialect of the programming language called ECMAScript. ECMAScript is maintained by the standards body Ecma International and the current ratified revision is version 5. All modern browsers provide great support for ECMAScript version 5 (abbreviated frequently as ES5) and Internet Explorer 9 has one of the best in class implementations from a standards compliance perspective.

Where does HTML5 come in?

How does JavaScript fit in with HTML5? It turns out that a vast majority of

the APIs being put out as part of the HTML5 specification are in fact JavaScript APIs. Canvas, GeoLocation, IndexedDB, WebSockets, Web Workers and so forth are all JavaScript APIs, i.e., JavaScript is the primary mechanism by which you programmatically interface with these APIs. Today JavaScript and HTML5 essentially go hand in hand and the term "HTML5" is a bit of a misnomer because it is no longer just about markup!

How do we go about using it?

Using JavaScript in HTML5 web pages couldn't be simpler. JavaScript can be used in HTML documents in two ways:

Inline scripts

It is possible to interleave HTML markup with JavaScript. Here's an example:

```
<!doctype html>
<html>
<head>
    <title>JavaScript and HTML5</title>
</head>
<body>
    <script>
        document.write("Hello, world!");
    </script>
</body>
</html>
```

The highlighted line is a simple line of JavaScript that uses the Document Object Model (DOM) interface to write a line of text to the document. You'll note that the "script" tag no longer requires a "type" or a "language" attribute to indicate that the language being used is JavaScript.

Include scripts

An alternative to embedding script inside HTML documents inline is to write them in separate files and then include them from the HTML file. An example follows:

hello.htm

```
<!doctype html>
<html>
<head>
    <title>JavaScript and HTML5</title>
```

```
    <script src="hello.js"></script>
</head>
<body>
</body>
</html>

hello.js
window.addEventListener("load", function () {
    document.write("Hello, world again!");
});
```

Here the script has been written in a separate file called "hello.js" and has been included from "hello.htm" via a "script" tag. As soon as the browser encounters a referenced script it immediately downloads the script and runs it. In this case the script simply adds an event handler for the "window" object's "load" event and adds a line of text to the document.

JavaScript logging and debugging

During development it is fairly common to encounter situations where we need to add diagnostic output to our programs – especially during the process of debugging. The case is no different with JavaScript. Most modern browsers these days include a "console" object as part of the JavaScript runtime that can be used to log diagnostic output to a debugger console. You typically use this object like so:

```
console.log("Diagnostic output here.");
```

Where does this output show up? In the debugger's output console. In most modern browsers today you can hit the F12 key on the keyboard to launch the developer tools included in the browser. Hitting F12 on Internet Explorer 9 to launch the debugger tool window.

JavaScript Object Notation

The last topic we'll examine in this section is JavaScript Object Notation (JSON). JSON is a simple human readable textual data interchange format used by web applications primarily when interfacing with AJAX based APIs. JSON is directly supported as part of the ES5 specification and again, all modern browsers support JSON serialization and deserialization out of the box. The terse hierarchical nature of the JSON syntax lends itself well to many scenarios that had in the past been the domain of XML. If you wanted to define a "person" object for instance with a few properties, here's how you would express it using JSON syntax:

```
var person = {
    "name": "Foo",
    "age": 10,
    "gender": "female",
    "address": {
        "street": "street 1",
        "city": "city 1",
        "pincode": 12345
    }
};
```

You'll note that we are able to nest objects inside objects as we have done above when defining the "address" object as a member of the "person" object. Browsers support the JSON API which allows you to serialize and deserialize objects to and from strings. Given the "person" object we defined above here's how you'd produce a string from it:

```
var str = JSON.stringify(person);
```

And here's how you'd convert that string right back into a "person" object:

```
var person2 = JSON.parse(str);
```

With the JSON API handy it becomes rather trivial to send and receive data to and from web servers.

CROSS BROWSER COMPATIBILITY

The browser market has a wide range of choices. In order to ensure your web site has the best experience, you need to optimise it across all browsers!

Introduction

Building web applications that provide a good user experience regardless of what browser the user is using to visit the site therefore becomes a critical aspect of web development. This situation is aggravated by the fact that not only do web developers have to worry about different brands of browsers (IE, Firefox, Chrome and Safari) they also have to deal with different versions of the same browser. While this might sound like a sad state of affairs, fortunately there are a few techniques that we can employ to deal with this issue.

Polyfills and Shims

A common refrain that one encounters with web developers is the idea that HTML5 is mostly just hype because of the ground reality that browser usage

is extremely fragmented. The only practical thing, it appears, is to target the lowest common denominator. This is not so. Two concepts that are worth discussing in this context are "polyfills" and "shims"!

A "polyfill" is basically just a fancy name for a library of some kind that provides the functionality of a particular HTML5 specification for a browser that does not natively provide that support. For example, older versions of Internet Explorer - up until version 8 - do not support the use of the HTML5 "canvas" tag. If being able to use that functionality in a page is critical to the app then it might make sense to use a polyfill library that provides the same functionality through some other means – perhaps via Silverlight or Flash plugins. The user experience will certainly not be the same as what it would be had the feature been natively supported but the users aren't completely out of luck either.

"Shims" are similar to polyfills in that they are libraries as well, except for the fact that they do not conform to any HTML5 specification. A good polyfill library would provide complete fidelity with the HTML5 spec in question (say, the Canvas API in the example cited before) where as a "shim" usually provides a workaround or provides a library that is at a higher level of abstraction. The "Amplify.js" library for instance solves the problem of client side storage using the best available option on a given browser. On browsers supporting the HTML5 DOM Storage or Indexed DB specs the library would leverage that and use alternatives on other browsers. A "shim" usually provides its own API for the functionality it provides and does not conform to any HTML5 spec even though it might leverage them when available.

Feature detection vs. Browser detection

It is a commonly used technique in web development these days to enable/disable functionality depending on the specific brand and/or version of browser a given user is running. This is usually achieved by inspecting the "user agent" string that browsers provide to identify themselves. In JavaScript for instance, it is common to find code such as the following:

```
if (navigator.userAgent.indexOf("MSIE") == -1) {
    window.addEventListener("load", onLoad);
} else {
    window.attachEvent("onload", onLoad);
}
```

This is not a good idea because here you are making assumptions about

what features are supported in a browser. In this particular case it turns out that from IE9 onwards the "addEventListener" method is in fact supported in the DOM. If you choose to do browser detection as we have done in the snippet above then it basically means that the onus of tracking what features are supported in what browser and in what version is upon you! A far better alternative is to use feature detection. The same functionality given above can be achieved using feature detection like so:

```
if (window.addEventListener) {
    window.addEventListener("load", onLoad);
} else if(window.attachEvent) {
    window.attachEvent("onload", onLoad);
}
```

In this case it really does not matter what browser or version the user is using, if "addEventListener" is available then it will be used. One library that really helps implement feature detection in an efficient manner is Modernizr which simplifies the process of testing for availability of various features. Testing whether canvas is supported for instance looks like this:

```
if (Modernizr.canvas) {
    // yep, canvas supported!
}
```

Internet Explorer 9 compatibility modes

Internet Explorer 9 displays webpages that contain the HTML5 document type in IE9 Standards mode, which provides the highest support available for established and emerging industry standards, including the HTML5 (Working Draft), W3C Cascading Style Sheets Level 3 Specification (Working Draft), Scalable Vector Graphics (SVG) 1.0 Specification, and others.

In some cases, existing websites may not display correctly when displayed in IE9 mode. This can happen for any number of reasons, including (but not limited to) the following:

- The design of a site may rely on the behavior of a specific version of a browser and that behavior has changed in later versions of the browser, such as conditional comments, version vectors, and user-agent detection.
- The design of a site may rely on non-standard behavior.
- The design of a site may rely on functionality that is no longer supported in the latest versions of various standards or browsers.

The design of a site may rely on functionality that was incorrectly implemented in an earlier version of a given browser.

To help ensure that your sites are available while you redesign them, Internet Explorer 9 supports document compatibility, which refers to a set of features that allow you to direct Internet Explorer 9 to display your pages as if they were viewed by older versions of the browser.

Document compatibility controls the features that are supported by webpages and the way webpages are displayed in the browser. An extension of the compatibility mode introduced in Microsoft Internet Explorer 6, document compatibility enables you to choose the specific rendering mode that Internet Explorer uses to display your webpages.

Specifying Document Compatibility Modes

You can use document modes to control the way Internet Explorer interprets and displays your webpage. To specify a specific document mode for your webpage, use the meta element to include an "X-UA-Compatible" header in your webpage, as shown in the following example.

```
<html>
<head>
  <!-- Enable IE9 Standards mode -->
  <meta http-equiv="X-UA-Compatible" content="IE=9" >
  <title>My webpage</title>
</head>
<body>
  <p>Content goes here.</p>
</body>
</html>
```

If you view this webpage in Internet Explorer 9, it will be displayed in IE9 mode. The following example specifies EmulateIE7 mode.

```
<html>
<head>
  <!-- Mimic Internet Explorer 7 -->
  <meta http-equiv="X-UA-Compat-
ible" content="IE=EmulateIE7" >
  <title>My webpage</title>
</head>
<body>
  <p>Content goes here.</p>
</body>
</html>
```

In this example, the X-UA-Compatible header directs Internet Explorer to mimic the behaviour of Internet Explorer 7 when determining how to display the webpage. This means that Internet Explorer will use the <!doctype> directive (or lack thereof) to choose the appropriate document type. Because this page does not contain a <!doctype> directive, the example would be displayed in IE5 (Quirks) mode.

The content attribute specifies the mode for the page; to mimic the behavior of Internet Explorer 7, specify IE=EmulateIE7. Specify IE=5, IE=7, IE=8, or IE=9 to select one of those compatibility modes. You can also specify IE=edge to tell Internet Explorer to use the highest mode available.

HTML5 CANVAS & SVG

The most noticeable difference in HTML5 is the way it handles and addresses concerns with image and video handling

We'll now explore what you can do with the HTML5 Canvas API – a cool API that enables you to dynamically generate and render graphics, charts, images, and animation. We'll walk through the basics of the rendering API to create a drawing, exploring all the features that Canavas boasts of, while also checking for Browser support and creating fallback contents. We will briefly review SVG and contrast the two technologies.

Introduction

As per the specifications of HTML5, the <canvas> element is defined as "a resolution-dependent bitmap canvas which can be used for rendering graphs, game graphics, or other visual images on the fly." A canvas is a rectangle in your page. When you use a canvas element in your web page, it creates a rectangular area on the page where you can use JavaScript to draw anything you want. By default, this rectangular area is 300 pixels

wide and 150 pixels high, but you can specify the exact size and set other attributes for your canvas element.

The HTML5 Canvas element is similar to tags like , <a>, and <table> tag, the exception being that its contents are rendered with JavaScript. In other words you will have to place the canvas tag somewhere inside your HTML, create a JS function that will access the canvas tag, and then utilize the HTML5 Canvas API to draw your visualizations.

Browser Support

Almost all browsers now provide support for HTML5 Canvas. However, there are some parts of the specification that were added later that are not supported as widely.

Versions of Internet Explorer before 9.0 do not support the canvas API. If you are using one of those earlier versions of Internet Explorer, you can use a Microsoft-proprietary technology called VML, which can do many of the same things as the <canvas> element. Or you could use ExplorerCanvas - excanvas.js which is an open source, Apache-licensed JavaScript library that implements the canvas API in earlier versions of Internet Explorer and other browsers that don't support canvas

To use explorercanvas, you should check if Canvas element is supported or not script tag in your web pages to and launch explorercanvas methods,

This technique is called as "feature detection" and is better than browser detection, as we can now be protected from browser support issues.

Universal support for the Canvas is highly desired by developers, and new projects are being launched all the time to try to bring support for Canvas to older or nonstandard browser environments.

Like all of the graphics in IE9, canvas is hardware accelerated through Windows and the GPU.

Fallback Content

In case your web page is accessed by a browser that does not support the canvas element or a subset of the HTML5 Canvas API features, it is a good idea to provide some alternate source. For example, you can just provide an alternate image or just some text that explains what the user could be enjoying if they actually used a modern browser.

```
<canvas>
  Please update your browser in order to enjoy canvas!
```

```
</canvas>
```

Browsers that do not support the canvas element will simply render this fallback text content. Instead of the text shown, you can also point to an image that can be displayed in case the browser does not support the canvas element.

CSS and Canvas

As with most HTML elements, CSS can be applied to the canvas element itself to add borders, padding, margins, etc. Some CSS values, like the font, are inherited by the contents of the canvas itself. As a result the fonts that are drawn into a canvas will default to the settings of the canvas element itself. Properties set on the context used in canvas operations follow similar syntax as CSS. Colors and fonts, for example, use the same notation on the context that they use throughout any HTML or CSS document.

HTML5 Canvas APIs

We'll explore the use of the HTML5 Canvas APIs in more detail. We will try to illustrate the various capabilities of HTML5 Canvas in a reasonable order.

Adding a Canvas to a Page

As you will admit, a blank canvas is pretty boring. So let us start drawing something.

```
<script>
  function drawDiagonal() {
    // Get the canvas element and its drawing context
    var canvas = document.getElementById('diagonal');
    var context = canvas.getContext('2d');

    // Create a path in absolute coordinates
    context.beginPath();
    context.moveTo(60, 120);
    context.lineTo(120, 60);

    // Stroke the line onto the canvas
    context.stroke();
  }

   window.addEventListener("load", drawDiagonal, true);
</script>
```

This JavaScript Code will create a diagonal line. Let us see how this code works:

- First of all you gain access to the canvas object by referencing a particular canvas's ID value. In this example, the ID is diagonal.
- Next, you create a context variable and you call the canvas object's getContext method, passing in the type of canvas you are looking for. You pass in the string "2d" to get a two-dimensional context.
- You then use the context to perform drawing operations. In this case, you can create the diagonal line by calling three methods—beginPath, moveTo, and lineTo—passing in the start and end coordinates of the diagonal.
- The moveTo() method creates a new subpath for the given point which becomes the new context point. You can think of the moveTo() method as a way to position your drawing cursor.
- The lineTo() method draws a line from the context point to the given point.
- The drawing methods moveTo and lineTo do not actually create the line; you finalize a canvas operation and draw the line by calling the context.stroke(); method
- The stroke() method assigns a color to the line and makes it visible. Unless otherwise specified, the default stroke color is black.

Almost all the important functions with visual output are accessible only from the context. This flexibility enables the canvas to support different types of drawing models in the future, based on the type of context that is retrieved from the canvas.

As we saw in the previous example, many operations on the context do not immediately update the drawing surface. Functions such as beginPath, moveTo, and lineTo do not modify the canvas appearance immediately. The same is true of many functions that set the styling and preferences of the canvas. Only when a path is stroked or filled does it appear on the display. Otherwise, the canvas will only be immediately updated when images are displayed or rectangles are drawn, filled, or cleared.

Applying Transformations

Now let's look at another way to draw on the canvas using transformation. In the following example, the result is identical to the previous example, but the code used to draw the diagonal line is different. Transformation can be thought of as a best practice for more complex canvas operations as it is critical to understanding the HTML5 Canvas API's complex capabilities.

```
<script>
function drawDiagonal() {
                var canvas = document.
getElementById('diagonal');
                var context = canvas.getContext('2d');

                 // Save a copy of the current drawing
state
                 context.save();

                // Move the drawing context to the right,
and down
                context.translate(70, 140);

                // Draw the same line as before, but using
the origin as a start
                context.beginPath();
                context.moveTo(60,120);
                context.lineTo(60, 120);
                 context.stroke();

                // Restore the old drawing state
                 context.restore();
       }

     window.addEventListener("load", drawDiagonal, true);

</script>
```

Let's examine the JavaScript code used to create this second, translated diagonal line. First, you access the canvas object by referencing its ID value (in this case, diagonal).

You then retrieve a context variable by calling the canvas object's getContext function. Next, you save the context so that you can get back to its original state at the end of the drawing and transformation operation. Saving lets us ensure that whatever modifications we make during the operation (translate, scale, and so on) will not be applied to the context in future operations.

We now apply the translate method to the context. The translation coordinates you supply will be added to the eventual drawing coordinates

(the diagonal line) at the time any drawing is rendered, but only after the drawing operation is complete.

After the translation has been applied, you can perform the normal drawing operations to create the diagonal line and render it to the canvas by calling the context.stroke method. Finally, you restore the context to its clean original state, so that future canvas operations are performed without the translation that was applied in this operation.

Canvas Coordinates

The coordinates in a canvas start at x=0, y=0 in the upper-left corner—which is referred to as the origin—and increase (in pixels) horizontally over the x-axis and vertically over the y-axis. Along the X-axis, values increase towards the right edge of the canvas. Along the Y-axis, values increase towards the bottom edge of the canvas.

The coordinate diagram can be drawn with a <canvas> element. It will comprise

- a set of off-white vertical lines
- a set of off-white horizontal lines
- two black horizontal lines
- two small black diagonal lines that form an arrow
- two black vertical lines
- two small black diagonal lines that form another arrow
- the letter "x"
- the letter "y"
- the text "(0, 0)" near the upper-left corner
- the text "(500, 375)" near the lower-right corner
- a dot in the upper-left corner, and another in the lower-right corner

First, we need to define the <canvas> element itself. The <canvas> element defines the width and height, and the id so we can find it later. Then we need a script to find the <canvas> element in the DOM and get its drawing context.

```
<canvas id="CoodSys" width="500" height="375"></canvas>
var CoodSys _canvas = document.getElementById("CoodSys
");
var context = CoodSys _canvas.getContext("2d");
```

Drawing Paths

Paths in the HTML5 Canvas API represent any shape you care to render. Our original line example was a path and we can complicate it as we desire,

with multiple line and curve segments and even subpaths.

When embarking on any routine to draw a shape or path, the first call you make is beginPath. This simple function takes no arguments, but it signals to the canvas that you wish to start a new shape description.

To draw straight lines in pencil, you use the following two methods:

1. moveTo(x, y) moves the pencil to the specified starting point.
2. lineTo(x, y) draws a line to the specified ending point.

The next special pathing function is a call to closePath. This command informs the canvas that the current shape has closed or formed a completely contained area.

Let's begin by drawing the off-white grid.

```
function createPath(context) {
// Draw the off-white grid
      context.beginPath();

      // Draw the vertical lines
for (var x = 0.5; x < 500; x += 10)
 {
context.moveTo(x, 0);
context.lineTo(x, 375):
}
// Draw the horizontall lines
for (var y = 0.5; y < 500; y += 10)
 {
context.moveTo(y, 0);
context.lineTo(y, 375):
}
context.strokeStyle = "#eee";
context.stroke();

       // Close the path back to its start point
      context.closePath();
}
```

Now we will draw the arrows. Since we want to draw the arrow in a different color ink — black instead of off-white — we will need to start a new path.

```
context.beginPath();

```

```
//horizontal arrow
context.moveTo(0, 40);
context.lineTo(240, 40);
context.moveTo(260, 40);
context.lineTo(500, 40);
context.moveTo(495, 35);
context.lineTo(500, 40);
context.lineTo(495, 45);

//vertical arrow
context.moveTo(60, 0);
context.lineTo(60, 153);
context.moveTo(60, 173);
context.lineTo(60, 375);
context.moveTo(65, 370);
context.lineTo(60, 375);
context.lineTo(55, 370);

//Change the stroke color to black.
context.strokeStyle = "#000";
context.stroke();

context.closePath();
```

Other features or context properties available with paths include:

- Increasing the line width
- Rounding the corners at path joints
- Specifying how the lines should display at end points.
- Setting the fill color.
- Drawing curves

Adding Text

In addition to drawing paths on a canvas, you can also draw text on a canvas. You can set a few font attributes, and then you pick a point on the canvas and draw your text there.

The following font attributes are available on the drawing context:

- font includes font style, font variant, font weight, font size, line height, and font family.
- textAlign controls text alignment. Possible values are start, end, left, right, and center.
- textBaseline controls where the text is drawn relative to the starting

point. Possible values are top, hanging, middle, alphabetic, ideographic, or bottom.

As you might expect, the text drawing routines consist of two functions on the context object:

1. fillText (text, x, y, maxwidth)
2. strokeText (text, x, y, maxwidth)

We can now draw the text we need:

```
context.font = "bold 12px sans-serif";
context.fillText("x", 248, 43);
context.fillText("y", 58, 165);

context.textBaseline = "top";
context.fillText("( 0 , 0 )", 8, 5);

context.textAlign = "right";
context.textBaseline = "bottom";
context.fillText("( 500 , 375 )", 492, 370);

context.fillRect(0, 0, 3, 3);
context.fillRect(497, 372, 3, 3);
```

Let's see what we have done here:

The first three lines are easy to understand. We have specified the font, the font size and font style and then we have written the letters 'x' and 'y'.

For the text in the upper-left corner, we have put the top of the text to be at y=5.Instead of measuring the height of the text and calculating the baseline, we have set textBaseline to top and passed the upper-left coordinate of the text's bounding box.

For the text in the lower-right corner, we have put the text to be at coordinates (492,370). Instead of measuring the height and the width of the text, we have set textAlign to right and textBaseline to bottom, and then called fillText() with the bottom-right coordinates of the text's bounding box.

The last two lines add two small rectangles and fill it with the default black color. Voila! We have our final coordinate system.

Now let us take a look at some other cool stuff that you can do with Canvas

Inserting Images

Images can be very easily displayed and manipulated inside a canvas. You can stamp or stretch an image, use transformations on it, and even make your image he focus of the entire canvas. The HTML5 Canvas API includes a few simple commands for adding image content to the canvas.

However keep in mind that images can also add a complication to the canvas operations as we have to wait for them to load. Browsers will usually be loading images asynchronously as your page script is rendering. However, if you attempt to render an image onto a canvas before it has completely loaded, the canvas will fail to render any image at all. As such, you should be careful to make sure the image is loaded completely before you attempt to render it.

The canvas drawing context defines a drawImage() method for drawing an image on a canvas. The method can take three, five, or nine arguments.

- drawImage(image, dx, dy) takes an image and draws it on the canvas. The given coordinates (dx, dy) will be the upper-left corner of the image. Coordinates (0, 0) would draw the image at the upper-left corner of the canvas.
- drawImage(image, dx, dy, dw, dh) takes an image, scales it to a width of dw and a height of dh, and draws it on the canvas at coordinates (dx, dy).
- drawImage(image, sx, sy, sw, sh, dx, dy, dw, dh) takes an image, clips it to the rectangle (sx, sy, sw, sh), scales it to dimensions (dw, dh), and draws it on the canvas at coordinates (dx, dy).

As we can see, in order to draw an image on a canvas, first of all we need an image. The image can be an existing <img> element, or you can create an Image() object with JavaScript. Either way, you need to ensure that the image is fully loaded before you can draw it on the canvas.

If you're using an existing <img> element, you can safely draw it on the canvas during the window.onload event. If you're creating the image object entirely in JavaScript, you can draw the image on the canvas during the Image.onload event.

Using Gradients

Gradients can be applied in the form of a gradual algorithmic sampling of colors. You can employ two different styles to do the same - stroke style or fill style. Creating gradients is a three-step process:

1. Create the gradient object itself.
2. Apply color stops to the gradient object, signaling changes in color along

the transition.

3. Set the gradient as either a fillStyle or a strokeStyle on the context.

It is perhaps easier to think of gradients as a smooth change of color that moves along a line. For example, if you supply points A and B as the arguments to the creation of a gradient, the color will be transitioned for any stroke or fill that moves in the direction of point A to point B.

The canvas drawing context supports two types of gradients:

1. createLinearGradient(x0, y0, x1, y1) paints along a line from (x0, y0) to (x1, y1).
2. createRadialGradient(x0, y0, r0, x1, y1, r1) paints along a cone between two circles. The first three parameters represent the start circle, with origin (x0, y0) and radius r0. The last three parameters represent the end circle, with origin (x1, y1) and radius r1.

Let us make a linear gradient

```
<canvas id="d" width="300" height="225"></canvas>
var d_canvas = document.getElementById("d");
var context = d_canvas.getContext("2d");
var my_gradient = context.createLinearGradient(0, 0, 300, 0);
my_gradient.addColorStop(0, "black");
my_gradient.addColorStop(1, "white");
context.fillStyle = my_gradient;
context.fillRect(0, 0, 300, 225);
```

Let us demystify this code block:-

- First of all we create an empty canvas of the size 300 X 225.
- Next we find the <canvas> element and access its drawing context
- Now we can start making the gradient. We make the gradient 300 pixels wide, like our canvas. Because the y values (the 2nd and 4th parameters) are both 0, this gradient will shade evenly from left to right.
- Once we have a gradient object, we can define the gradient's colors. A gradient has two or more color stops, which can be anywhere along the gradient. To add a color stop, you need to specify its position along the gradient and it can take any value between 0 to 1.Our gradient shades from black to white
- To render the gradient, you set your fillStyle to the gradient and draw a shape, like a rectangle or a line.

Presto! We have our gradient.

Suppose we want a gradient that shades from top to bottom. When you create the gradient object, keep the x values (1st and 3rd parameters) constant, and make the y values (2nd and 4th parameters) range from 0 to the height of the canvas.

```
var my _ gradient = context.createLinearGradient(0, 0, 0, 225);
my _ gradient.addColorStop(0, “black”);
my _ gradient.addColorStop(1, “white”);
context.fillStyle = my _ gradient;
context.fillRect(0, 0, 300, 225);
```

Pattern

The HTML5 Canvas API also includes an option to set an image as a repeatable pattern for either a path stroke or fill. To create a pattern with the HTML5 Canvas, we can define a pattern with the createPattern (), set it to the fill style using the fillStyle property, and then fill a shape using fill (). The createPattern () method requires an image object and a repeat option, which can be repeat, repeat-x, repeat-y, or no-repeat.

What you don’t see

Pixel Data

One of the most useful—albeit nonobvious—aspects of the Canvas API is the ability for developers to easily access the underlying pixels in the canvas. This access is both a good and a bad thing: it is easy to get access to the pixel value, and it is equally easy to modify those values and apply them back to the canvas. In fact, you can manipulate the canvas entirely through the pixel value calls and forget all about the different rendering calls we’ve discussed in this chapter. This is made possible by the existence of three functions on the context API –

- context.getImageData(sx, sy, sw, sh)
- context.putImageData(imagedata, dx, dy)
- context.createImageData(sw, sh)

Canvas Security

Although most developers would use pixel manipulation for legitimate means, it is quite possible that the ability to fetch and update data from a canvas could be used for not-so-legitimate purposes as well. For this reason, the concept of an origin-clean canvas was specified. What it does is that for any canvas tainted with images from origins other than the source of the

containing page, will not have their data retrieved.

Before the arrival of the Canvas API, it was not possible to programmatically retrieve the pixel values of a downloaded image. Private images from other sites could be displayed in a page but not read or copied. Allowing scripts to read image data from other origins would effectively share users' photographs and other sensitive online image file with the entire web.

In order to prevent this, any canvas that contains images rendered from remote origins will throw a security exception if the getImageData function is called. It is perfectly acceptable to render remote images into a canvas from another origin as long as you do not attempt to fetch the data from that canvas after it has been tainted.

SVG

Scalable Vector Graphics (SVG) integrates with the HTML5 and CSS features to unleash some of the most beautiful experiences on the web.

The format makes use of specific XML tags to generate a vector-based image. This makes SVG more easily read and edited by hand than other graphic formats. And because an SVG file is composed entirely of markup, text within the vector image is searchable. Moreover, SVG benefits from the richness of having a DOM. Images are scriptable graphic objects, enabling much more powerful web applications. It's possible to create dynamically generated images without requiring server-side scripting or browser plug-ins.

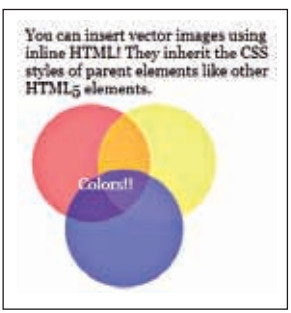
SVG offers many distinct advantages over other formats. SVG images scale up with little increase in file size and with no loss of fidelity, making it ideal for use in smaller images and icons. SVG is also a fantastic format for images with larger dimensions. Maps, charts, and diagrams are examples. Small labels and detailed borders remain intact even when images are scaled down. Key areas of focus, like the text in an organization chart or the border of a country, remain crisp and readable after zooming in on the image. The preservation of information in human-readable markup also makes SVG suitable for Wiki-type collaborative images, as it is easily edited. SVG makes changing labels in charts and diagrams a breeze – only a basic text editor is needed. As native SVG support in web browsers continues to broaden, SVG becomes an increasingly good option for implementing interactive and animated web content.

In HTML5, SVG can be included as inline HTML. Current webpages can be updated to include inline SVG with little alteration to their overall

structure! For example, look at the following simple, hand-written markup:

```
<!DOCTYPE html>
<p style="font-family:Georgia;font-size:9pt;">You can insert vector images using inline HTML! They inherit the CSS styles of parent elements like other HTML5 elements.
<BR>
<svg width="200" height="100">
    <circle cx="50" cy="50" r="45" fill-opacity=".5" fill="red"/>
    <circle cx="100" cy="50" r="45" fill-opacity=".5" fill="yellow"/>
    <circle cx="75" cy="100" r="45" fill-opacity=".5" fill="blue"/>
    <text x="40" y="70" fill="white">Colors!!</text>
</svg>
</p>
```

IE9 renders it as:



In addition to creating images from scratch, you can use SVG to mark up existing images without editing the underlying image. The following SVG attempts to point out a couple of tough-to-spot otters:

```
<!DOCTYPE html>
<svg width="120" height="90" >
    <style>
        .highlight {
            stroke-width: 5px;
            stroke: white;
            fill: none;
        }
    </style>

    <image x="0" y="0" width="120" height="90" xlink:href="http://photos-a.ak.fbcdn.net/photos-ak-sf2p/v160/140/48/1107073/n1107073_31705640_3439.jpg"/>
    <circle cx="90" cy="50" r="15" class="highlight"/>
    <circle cx="22" cy="47" r="10" class="highlight"/>
    <rect x="0" y="0" width="120" height="20" fill-opacity=".5" fill="black"/>
```

```
    <text x="5" y="15" fill="white" font-size="8pt" font-
family="Verdana">we saw giant otters!</text>
    <rect x="0" y="0" width="120" height="90" stroke-
width="1" stroke="black" fill="none"/>
</svg>
```

High Level Contrast of Canvas and SVG

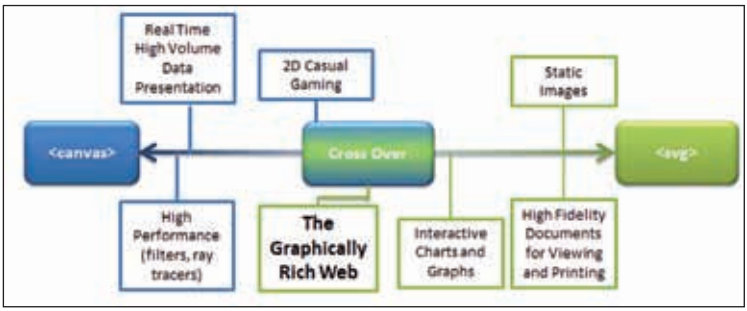
The following is a high-level contrast of Canvas and SVG meant to frame a discussion of when to use one particular vector graphic technology over the other.

A COMPARISON OF CANVAS AND SVG	
Canvas	SVG
Pixel-based (canvas is essentially an image element with a drawing API)	Object Model-based (SVG elements are similar to HTML elements)
Single HTML element similar to <img> in behavior	Multiple graphical elements which become part of the Document Object Model (DOM)
Visual presentation created and modified programmatically through script	Visual presentation created with markup and modified by CSS or programmatically through script
Event model/user interaction is coarse—at the canvas element only; interactions must be manually programmed from mouse coordinates	Event model/user interaction is object-based at the level of primitive graphic elements—lines, rectangles, paths
API does not support accessibility; markup-based techniques must be used in addition to canvas	SVG markup and object model directly supports accessibility

SVG is known as a retained mode graphics model persisting in an in-memory model. Analogous to HTML, SVG builds an object model of elements, attributes, and styles. When the <svg> element appears in an HTML5 document, it behaves like an inline block and is part of the HTML document tree.

Canvas is a bitmap with an immediate mode graphics application programming interface (API) for drawing on it. Canvas is a "fire and forget" model that renders its graphics directly to its bitmap and then subsequently has no sense of the shapes that were drawn; only the resulting bitmap stays around.

One way to think of these is that Canvas resembles the Windows GDI API, where you programmatically draw graphics to a window, and SVG



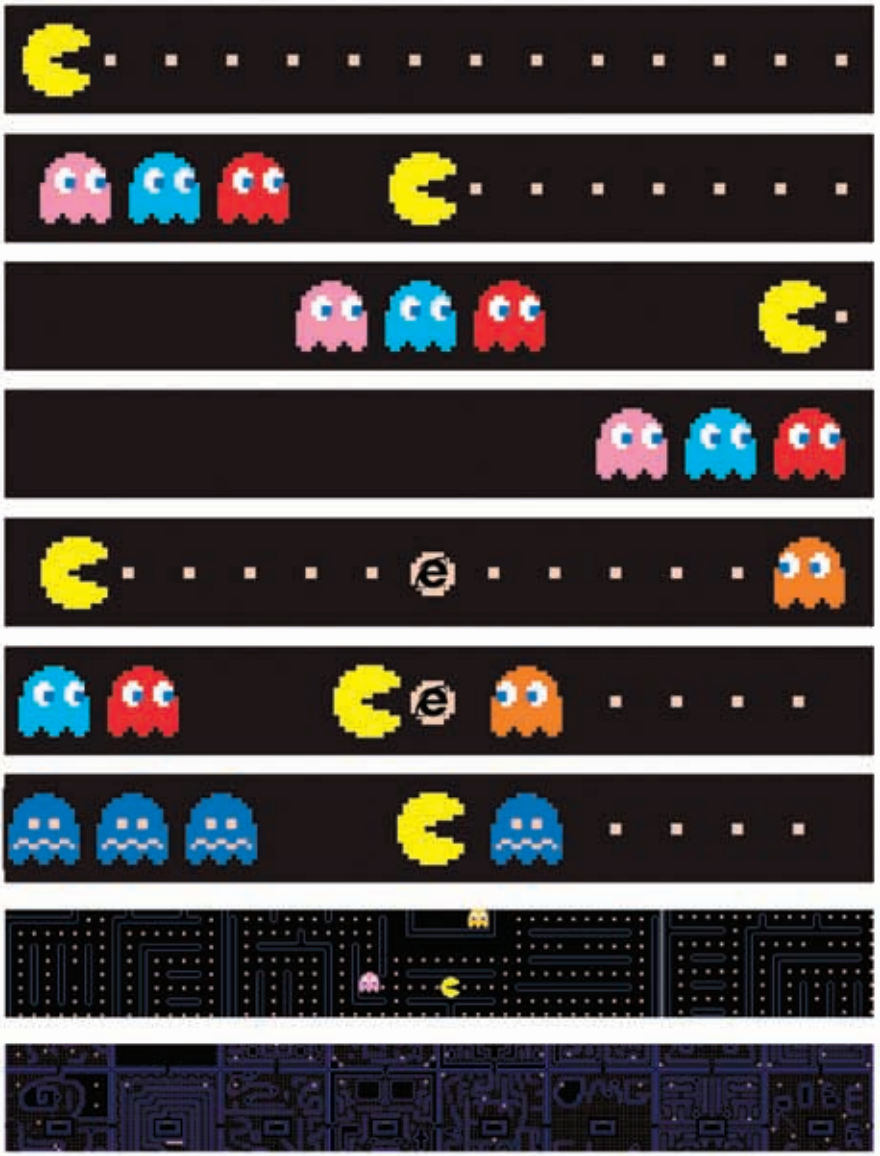
resembles HTML markup with elements, styles, events, and DOM-based programmability. Canvas is procedural whereas SVG is declarative.

The analysis of existing vector graphic technologies available in the latest modern browsers shows new scenarios can be created using standard Web technologies in an interactive way.

The ongoing evolution of the Web continues to include richer graphics at its heart. We've presented one point-of-view about applying these technologies to particular scenarios. In the end, both Canvas and SVG are important components of the HTML5 graphically rich Web.

Summary

As you can see, the HTML5 Canvas API and SVG provides a very powerful way to modify the appearance of your web application without resorting to odd document hacks. Images, gradients, and complex paths can be combined to create nearly any type of display you may be looking to present. If you learn to harness the power of the canvas and SVG, you can create many different applications that were never possible in a web page before, like graphs, charts, image editing, and so on.



Hardware accelerated graphics. It's like Power Pellets for the web.

THE WORLD'S BIGGEST PAC-MAN

Over one billion dots eaten.
Built for Internet Explorer 9.

Windows Internet Explorer 9

http://worldsbiggestpacman.com

HTML5 – VIDEO AND AUDIO

HTML5 has drastically simplified the way multimedia content is addressed and referenced.

We'll explore what you can do with two of the most important HTML5 elements – audio and video. The audio and video elements add new media options to HTML5 applications that allow you to use audio and video without plugins and at the same time provides a common, integrated, and scriptable API.

Multimedia over web until today was delivered via plug-ins such as Adobe Flash, Silverlight and Windows media player. While these plug-ins worked well, there has never been a standard for showing audio/video on a web page. A user always had to download a plug-in to watch content.

One of the most exciting new features of HTML5 is the inclusion of the <video> and <audio> elements, which allow developers to include multimedia content directly in their pages without the need for any plugin-based solution.

Audio/Video Containers

An audio or video file is nothing more than a container file, very similar to a ZIP archive file that contains a number of files. Just like a ZIP file can contain any sort of file within it, video container formats only define how to store things within them, not what kinds of data are stored.

A video container usually contains multiple tracks — a video track (without audio), plus one or more audio tracks (without video). These tracks are interrelated. An audio track contains markers within it to help synchronize the audio with the video. Individual tracks can have metadata, such as the aspect ratio of a video track, or the language of an audio track. Containers can also have metadata, such as the title of the video itself, cover art for the video, episode numbers (for television shows), and so on.

Overview of a Video Container

The figure below shows how a video file (a video container) contains audio tracks, video tracks, and additional metadata.

Some of the popular video container formats include the following:

- MPEG 4 (.mp4 or .m4v) - The MPEG 4 container is based on Apple's older QuickTime container (.mov).
- Flash Video (.flv) – This is the most common of all and is used by Adobe Flash. Earlier this was the only container format that Flash supported. More recent versions of Flash also support the MPEG 4 (.mp4) container.
- Ogg (.ogv) - It is an open standard, open source–friendly, and unencumbered by any known patents. The Ogg video is called "Theora" while the Ogg audio is called "Vorbis".
- WebM – This is a new container format announced in May, 2010. It is technically similar to another format, called Matroska (.mkv). It has been designed to be used exclusively with the VP8 video codec and Vorbis audio codec.
- Audio Video Interleave (.avi) - The AVI container format was invented by Microsoft in a simpler time, when the fact that computers could play video at all was an amazing thing.

Audio and Video Codecs

When you are "watching a video," your video player is doing at least three things at once:

1. Interpreting the container format to find out which video and audio tracks are available, and how they are stored within the file so that it can find the data it needs to decode next

2. Decoding the video stream and displaying a series of images on the screen
3. Decoding the audio stream and sending the sound to your speakers

The word "codec" is a portmanteau, a combination of the words "coder" and "decoder." Audio and video codecs are just algorithms used to encode and decode a particular audio or video stream, in a particular way. Raw media files can be enormous. Without encoding a video or audio clip, it would be too large to transmit across the Internet in a reasonable amount of time. Similarly without a decoder, the recipient would not be able to reconstitute the original media source from the encoded form. A codec is able to understand a specific container format and decodes the audio and video tracks that it contains.

A codec can be either lossy or lossless. A lossy video codec means that information is being irretrievably lost during encoding while lossless codec allows the exact original data to be reconstructed from the compressed data. Lossless codecs are much too big to be useful on the web and so only lossy codecs are used.

Common video codecs

H.264 - It is also known as "MPEG-4 part 10," a.k.a. "MPEG-4 AVC," a.k.a. "MPEG-4 Advanced Video Coding." It aims to provide a single codec for low-bandwidth, low-CPU devices (cell phones); high-bandwidth, high-CPU devices (modern desktop computers); and everything in between.

THEORA – This is a royalty-free codec and is not encumbered by any known patents other than the VP3 patents, from which it evolved originally. Theora video can be embedded in any container format, although it is most often seen in an Ogg container.

VP8 – This is another codec from On2, the same company that originally developed VP3 (later Theora). Technically, it produces output on par with H.264 High Profile, while maintaining a low decoding complexity on par with H.264 Baseline.

The most common audio codecs are:

- MPEG-1 Audio Layer 3 – You may know them as MP3. They can be encoded at different bitrates like 64 kbps, 128 kbps, 192 kbps, and so on. Higher bitrates mean larger file sizes and better quality audio. The MP3 format allows for variable bitrate encoding, which means that some parts of the encoded stream are compressed more than others.
- Advanced Audio Coding – AAC was designed to provide better sound quality than MP3 at the same bitrate, encode audio at any bitrate and up to 48 channels of sound. Like H.264 it also defines multiple profiles and it is patent encumbered.

- Vorbis - Vorbis audio streams are usually embedded in an Ogg or WebM container and support an arbitrary number of sound channels.

Audio and video restrictions

There are a few things that are not supported in the HTML5 audio and video specification:

HTML5 does not support audio and video streaming. That is, there is currently no standard for bitrate switching in HTML5 video; only full media files are supported by current implementations. However, there are aspects of the spec that are designed to support streaming media in the future once the formats are supported.

Media is restricted by HTTP cross-origin resource sharing.

Full-screen video is not scriptable because it could be considered a security violation to let a scriptable element take over the full screen. However, browsers have the option of letting users choose to view videos in full screen through additional controls.

Accessibility for audio and video elements is not fully specified yet. Work is underway on a specification called WebSRT for subtitle support based on the popular SRT format.

Browser support

As we can see there is no single combination of containers and codecs that works in all HTML5 browsers and it is not likely to change in the near future.

To make your video watchable across all of these devices and platforms, you're going to need to encode your video more than once –

1. Make one version that uses WebM (VP8 + Vorbis).
2. Make another version that uses H.264 baseline video and AAC "low complexity" audio in an MP4 container.
3. Make another version that uses Theora video and Vorbis audio in an Ogg container.
4. Link to all three video files from a single <video> element, and fall back to a Flash-based video player.

Using the HTML5 Audio and Video APIs

There are two main benefits to using these HTML5 media tags over previous video-embedding techniques—

With the new audio and video tags as a part of the native browser environment, it removes various deployment hurdles. Even though there are some

BROWSER SUPPORT FOR DIFFERENT CODECS

Browser	Details	Video Codec	Audio Codec
Internet Explorer	Version 9.0 and later	MP4 container, H.264 video,WebM, Baseline, Main, and High profiles	MP3, and AAC in MP4 container
Mozilla Firefox	Version 3.5 and later	Theora Videos.	Vorbis audio in an Ogg container
Google Chrome	Version 3.0 and later	Theora Videos.	Vorbis audio in an Ogg container
Safari	Version 3.0 and later	H.264 video (main profile)	AAC audio in an MP4 container.
Opera	Version 10.5 and later	Theora Videos.	Vorbis audio in an Ogg container
iPhone	Version 3.0 and later	H.264 video (baseline profile)	AAC audio ("low complexity" profile) in an MP4 container.
Android	Version 2.0 and later	H.264 video (baseline profile)	AAC audio ("low complexity" profile) in an MP4 container.

plugins with high install rates, they are often blocked in controlled corporate environments. Some users choose to disable these plugins due to the ostentatious advertising. Then there is security issues associated with some of them. Also sometimes it is difficulty for some plugins to integrate their displays with the rest of browser content - causing clipping or transparency issues.

The media elements expose a common, integrated, and scriptable API to the document. As a developer, your use of the new media elements allows very simple ways to script the control and playback of content. We will see multiple examples of this later in the chapter.

AUDIO

Let us first take a look at the Audio tag:

The Spec:

Currently, the HTML5 spec defines five attributes for the <audio> element:

1. src — a valid <abbr>URL</abbr> specifying the content source.
2. autoplay — a boolean specifying whether the file should play as soon as it can.
3. loop — a boolean specifying whether the file should be repeatedly played.
4. controls — a boolean specifying whether the browser should display its default media controls.
5. preload — none / metadata / auto — where 'metadata' means preload just the metadata and 'auto' leaves the browser to decide whether to preload the whole file.

   Incidentally these attributes are also defined for the <video> element.

   Let's create a simple example that will play an audio file:

<audio src="SultansofSwing.ogg" controls preload="auto" autobuffer></audio>

To create your own controls, you can use the API methods defined by the spec:

- Load () - Loads the media file and prepares it for playback.
- play() — plays the audio
- pause() — pauses the audio
- canPlayType() — interrogates the browser to establish whether the given mime type can be played
- buffered() — attribute that specifies the start and end time of the buffered part of the file

The source

The best way to force the browsers into playing audio is to use the <source> element. The browser will try to load the first audio source, and if it fails or isn't supported, it will move on to the next audio source. In addition, we can embed a Flash player if all else fails:

```
<audio controls preload="auto" autobuffer>
  <source src="elvis.mp3" />
  <source src="elvis.ogg" />
  <!-- now include flash fall back -->
</audio>
```

Cross-Browser Implementation

HTML5's <video>, <audio>, and <canvas> support fallback content. Meaning anywhere they are supported, they visually hide content placed beneath them. This allows you to target older browsers with a plug-in, or a link to alternate content. But as soon as a browser adds support for these elements, they automatically use them e.g.

```
<!-- Elements with fallback content -->
<audio id=vid1 controls width>
<source src="audio.mp3" type="audio/mp3" />
<source src="audio.ogg" type="audio/ogg" />
 <object type="application/x-silverlight-2"
              width="640" height="384">

  <a href="test.mp3">
   Download Audio
  </a>
```

```
 </object>
</Audio>
```

Or you could use Modernizer in your JavaScript to detect Audio support and Type support in a browser and render accordingly

```
var audio = new Audio();
audio.src = Modernizr.audio.ogg ? 'background.ogg' :
            Modernizr.audio.mp3 ? 'background.mp3' :
                                  'background.m4a';
audio.play();
```

VIDEO

Now let us take a look at the Video Tag –

1. The Markup

To make HTML video work in your site, the following lines should be sufficient.

```
<video>
  <source src="movie.mp4" type='video/mp4;
codecs="avc1.42E01E, mp4a.40.2"' />
  <source src="movie.webm" type='video/webm; codecs="vp8,
vorbis"' />
</video>
```

This snippet uses the <source> tag which lets you include multiple formats as fallback types in case the user's browser doesn't support one of them. You also need to keep in mind –

- Ensure that your server is serving video files with the correct MIME type in the Content-Type header.
- If your app is being served in a Google App Engine app then you would need to add something like the following to app.yaml (one for each format):

```
  url: /(.*\.ogv)
static_files: videos_folder/\1
mime_type: video/ogg
upload: videos_folder/(.*\.ogv)
```

- Don't forget to specify the type attribute in the source tags when dealing with multiple formats to help the browser decide which format to download and play.

Video Formats

Almost all browser vendors, nowadays, support a common video format. You can make a snippet that has a combination of formats so you can be sure your video is displayed in all browsers:

```
<video>
  <source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />
  <source src="movie.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"' />
  Video tag not supported. Download the video <a href="movie.webm">here</a>.
<video>
```

Encoding

If you need to encode your existing videos to the formats mentioned in the previous section you can use any common converters that work well with both Windows and Mac, to easily get the format you need.

Attributes

Apart from the 5 common attributes with the <audio> tag, the video tag has the following attributes –

1. Download – Offers to download the video instead, if the browser doesn't know what to do with video tag, or if there is a display error.
2. Autobuffer - Tells the player whether or not to attempt to load the media file before playback is initiated. If the media is set for auto-playback, this attribute is ignored.
3. Poster - You can use the poster attribute to display a frame of the video (as a .jpg, .png, whatever), in case the video doesn't load for some reason.
4. Audio - Specifies the default state of the audio.
5. Height - Sets the height of the video player
6. Width - Sets the width of the video player
5. Integration

As we said in the introduction, the main advantage of the video tag being a member of the HTML5 family, is the integration with the other layers of web development stack. Let us take a look at the kind of interesting stuff you can do

a. VIDEO + other HTML

All the common HTML attributes, we described in the previous section, can be used in the video player.

```
<video poster="star.png" autoplay loop controls tab-index="0">
  <source src="movie.webm" type='video/webm; codecs="vp8, vorbis"' />
```

```
  <source src="movie.ogv" type='video/ogg; codecs="theora,
vorbis"' />
</video>
```

The full level of integration of the video tag as a native browser element deprecates issues that might arise with third party embedded players such as drop down menus and iframes overlaying on the player or weird layout behavior when the other HTML elements surrounding the video were dynamically resized. Since the video is not treated as an embedded foreign object, there are some other benefits that affect the user experience. For instance, even if the focus is on the player itself actions like page scrolling or keyboard key strokes will be completely functional.

b. VIDEO + JS

The video tag comes with a set of attributes and methods which let you have a fine control of your video from your JS code. As any other HTML element you can attach common events to the video tag such as drag events, mouse events, focus events, etc. Since loading a video resource can have many caveats there are many events to have a fine control of the loading process, both at network level (loadstart, progress, suspend, abort, error, emptied, stalled) and at buffering level (loadedmetadata, loadeddata, waiting, playing, canplay, canplaythrough).

At its simplest level you can attach the canplay event to start doing stuff with the video.

```
video.addEventListener('canplay', function(e) {
  this.volume = 0.4;
  this.currentTime = 10;
  this.play();
}, false);
```

c. VIDEO + CSS

The video tag can also be styled using traditional CSS (e.g. border, opacity, etc) since it is a first-class citizen in the DOM. But the cool thing is that you can even style it with the latest CSS3 properties like reflections, masks, gradients, transforms, transitions and animations.

d. VIDEO + CANVAS

Canvas holds loads of possibilities when used in conjunction with the video tag. You can make use of the following features –

1. Import images - The drawImage method lets you import images from three

different sources: an image element, another canvas element and a <video> element!

2. Export Images - The toDataURL method lets you export the content of the canvas to an image
3. You can even hide the canvas you use to import/export.
4. You can increase the frequency at which you import and render the image from the video in order to emulate the same video frame rate in the canvas. This lets you do all sorts of fancy transforms in the canvas as if you were doing it in the video.
5. You can use the same technique of importing images to WebGL where you will be able, for instance, to import the frames of a video and render them on a spinning 3D cube.

e. VIDEO + SVG

SVG images and their behaviors are defined in XML text files. SVG provides a programmatic way to render and manipulate vector graphics and comes with features such as SVG filter effects. With these filters you can target a specific DOM element and apply some out of the box effects such as blur, composite, tiles, etc

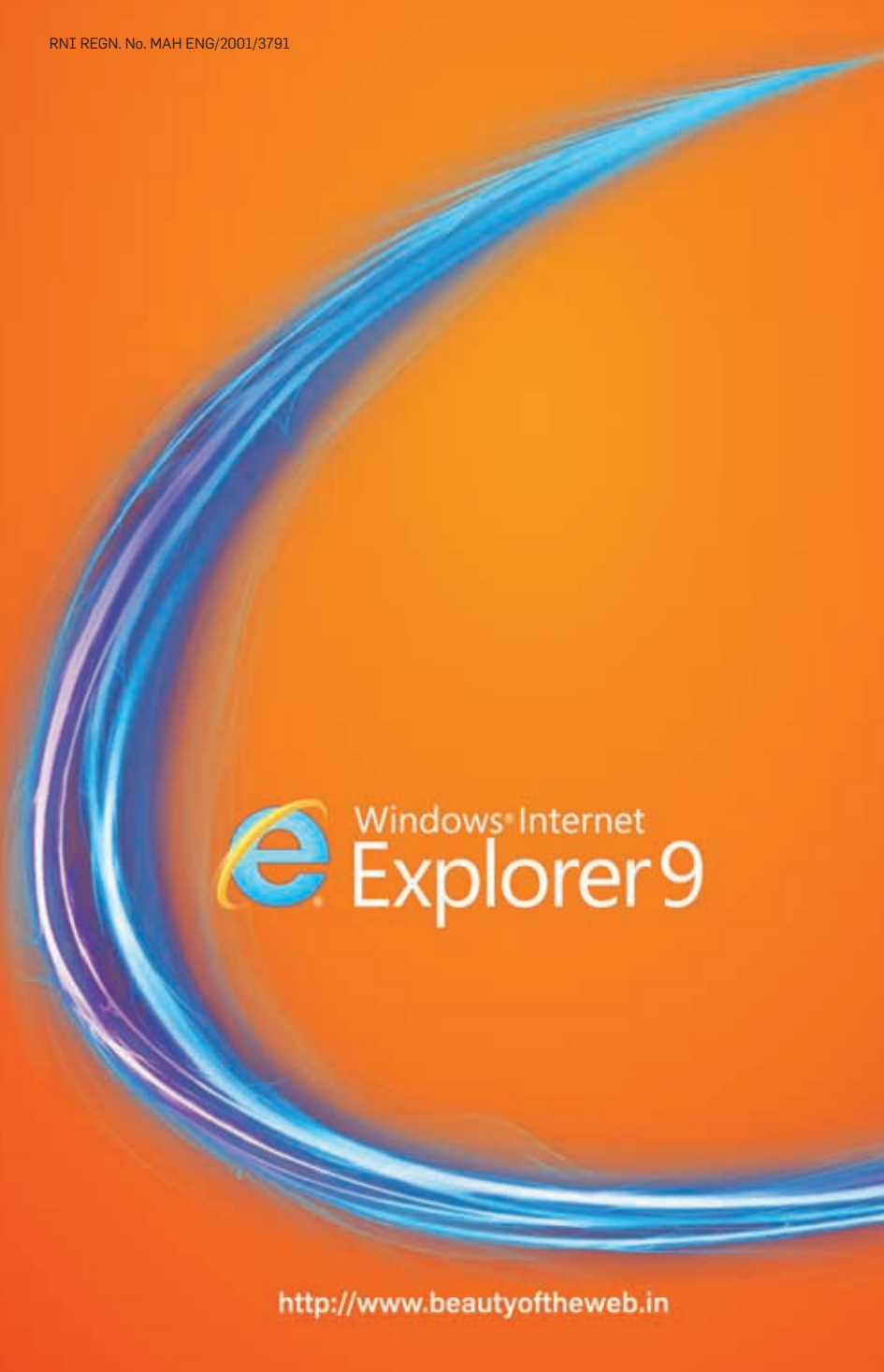
Summary

In this chapter, we have explored what you can do with the two important HTML5 elements audio and video. As we have seen the audio and video elements allow us to use audio and video without plugins, while at the same time providing a common, integrated, and scriptable API. We have seen what are the audio and video container files and codecs and how they are supported across different browsers. Next, we showed you how you can use control audio and video programmatically using the APIs and how to integrate these elements with other layers of web development stack.

Resources

http://windowsteamblog.com/ie/b/ie/
http://buildmypinnedsite.com/en
http://www.pinmywebsite.com/
http://www.zeollar.com/
http://blogs.msdn.com/b/ie/
http://ie.microsoft.com/testdrive/Default.html
http://www.beautyoftheweb.in

RNI REGN. No. MAH ENG/2001/3791



http://www.beautyoftheweb.in